2023-04-17_WorldDemographics-Test-02

May 15, 2023

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

%matplotlib inline
colors=plt.rcParams['axes.prop_cycle'].by_key()['color']
```

1 Exploration of UN world population data

available at: https://population.un.org/wpp/Download/Standard/CSV/

Available columns in dataset: * Total Population, as of 1 January (thousands) * Total Population, as of 1 July (thousands) * Male Population, as of 1 July (thousands) * Female Population, as of 1 July (thousands) * Population Density, as of 1 July (persons per square km) * Population Sex Ratio, as of 1 July (males per 100 females) * Median Age, as of 1 July (years) * Natural Change, Births minus Deaths (thousands) * Rate of Natural Change (per 1,000 population) * Population Change (thousands) * Population Growth Rate (percentage) * Population Annual Doubling Time (years) * Births (thousands) * Births by women aged 15 to 19 (thousands) * Crude Birth Rate (births per 1,000 population) * Total Fertility Rate (live births per woman) * Net Reproduction Rate (surviving daughters per woman) * Mean Age Childbearing (years) * Sex Ratio at Birth (males per 100 female births) * Total Deaths (thousands) * Male Deaths (thousands) * Female Deaths (thousands) * Crude Death Rate (deaths per 1,000 population) * Life Expectancy at Birth, both sexes (years) * Male Life Expectancy at Birth (years) * Female Life Expectancy at Birth (years) * Life Expectancy at Age 15, both sexes (years) * Male Life Expectancy at Age 15 (years) * Female Life Expectancy at Age 15 (years) * Life Expectancy at Age 65, both sexes (years) * Male Life Expectancy at Age 65 (years) * Female Life Expectancy at Age 65 (years) * Life Expectancy at Age 80, both sexes (years) * Male Life Expectancy at Age 80 (years) * Female Life Expectancy at Age 80 (years) * Infant Deaths, under age 1 (thousands) * Infant Mortality Rate (infant deaths per 1,000 live births) * Live births Surviving to Age 1 (thousands) * Deaths under age 5 (thousands) * Under-five Mortality Rate (deaths under age 5 per 1,000 live births) * Mortality before Age 40, both sexes (deaths under age 40 per 1,000 live births) * Male mortality before Age 40 (deaths under age 40 per 1,000 male live births) * Female mortality before Age 40 (deaths under age 40 per 1,000 female live births) * Mortality before Age 60, both sexes (deaths under age 60 per 1,000 live births) * Male mortality before Age 60 (deaths under age 60 per 1,000 male live births) * Female mortality before Age 60 (deaths under age 60 per 1,000 female live births) * Mortality between Age 15 and

50, both sexes (deaths under age 50 per 1,000 alive at age 15) * Male mortality between Age 15 and 50 (deaths under age 50 per 1,000 males alive at age 15) * Female mortality between Age 15 and 50 (deaths under age 50 per 1,000 females alive at age 15) * Mortality between Age 15 and 60, both sexes (deaths under age 60 per 1,000 alive at age 15) * Male mortality between Age 15 and 60 (deaths under age 60 per 1,000 males alive at age 15) * Female mortality between Age 15 and 60 (deaths under age 60 per 1,000 females alive at age 15) * Net Number of Migrants (thousands) * Net Migration Rate (per 1,000 population)

Each row corresponds to one country or region, and one year, for the interval 1950 to 2100 where future years contain projected data. The dataset contains rows for ”aggregated regions” (such as continents). We will later discard the projected data and the aggregated regions.

1.1 Import data

[2]:
```
# set up column data types
headerLine="SortOrder,LocID,Notes,ISO3_code,ISO2_code,SDMX_code,LocTypeID,LocTypeName,"+\
    "ParentID,Location,VarID,Variant,"+\
   ␣
  ↪"Time,TPopulation1Jan,TPopulation1July,TPopulationMale1July,TPopulationFemale1July,"+\
   ␣
  ↪"PopDensity,PopSexRatio,MedianAgePop,NatChange,NatChangeRT,PopChange,PopGrowthRate,"+\
   ␣
  ↪"DoublingTime,Births,Births1519,CBR,TFR,NRR,MAC,SRB,Deaths,DeathsMale,DeathsFemale,"+\
   ␣
  ↪"CDR,LEx,LExMale,LExFemale,LE15,LE15Male,LE15Female,LE65,LE65Male,LE65Female,"+\
    "LE80,LE80Male,LE80Female,InfantDeaths,IMR,LBsurvivingAge1,Under5Deaths,"+\
   ␣
  ↪"Q5,Q0040,Q0040Male,Q0040Female,Q0060,Q0060Male,Q0060Female,Q1550,Q1550Male,"+\
    "Q1550Female,Q1560,Q1560Male,Q1560Female,NetMigrations,CNMR"

dtypeDict={}
for x in headerLine.split(","):
    dtypeDict[x]=np.float64
for x in "SortOrder,LocID,LocTypeID,ParentID,VarID,Time".split(","):
    dtypeDict[x]=np.int32
for x in "Notes,ISO3_code,SDMX_code,ISO2_code,LocTypeName,Location,Variant".
  ↪split(","):
    dtypeDict[x]=str
```

[3]:
```
dataFull=pd.read_csv("data/WPP2022_Demographic_Indicators_Medium.
  ↪csv",sep=",",dtype=dtypeDict)
```

[4]:
```
# check list of columns; labels much shorter than above, but consistent
dataFull.keys()
```

```
[4]: Index(['SortOrder', 'LocID', 'Notes', 'ISO3_code', 'ISO2_code', 'SDMX_code',
        'LocTypeID', 'LocTypeName', 'ParentID', 'Location', 'VarID', 'Variant',
        'Time', 'TPopulation1Jan', 'TPopulation1July', 'TPopulationMale1July',
        'TPopulationFemale1July', 'PopDensity', 'PopSexRatio', 'MedianAgePop',
        'NatChange', 'NatChangeRT', 'PopChange', 'PopGrowthRate',
        'DoublingTime', 'Births', 'Births1519', 'CBR', 'TFR', 'NRR', 'MAC',
        'SRB', 'Deaths', 'DeathsMale', 'DeathsFemale', 'CDR', 'LEx', 'LExMale',
        'LExFemale', 'LE15', 'LE15Male', 'LE15Female', 'LE65', 'LE65Male',
        'LE65Female', 'LE80', 'LE80Male', 'LE80Female', 'InfantDeaths', 'IMR',
        'LBsurvivingAge1', 'Under5Deaths', 'Q5', 'Q0040', 'Q0040Male',
        'Q0040Female', 'Q0060', 'Q0060Male', 'Q0060Female', 'Q1550',
        'Q1550Male', 'Q1550Female', 'Q1560', 'Q1560Male', 'Q1560Female',
        'NetMigrations', 'CNMR'],
       dtype='object')
```

```
[6]: # confirm time range of data:
minyear,maxyear=[dataFull["Time"].min(),dataFull["Time"].max()]
print([minyear,maxyear])
```

```
[1950, 2101]
```

```
[7]: # how many rows and columns does the dataset have?
dataFull.shape
```

```
[7]: (43472, 67)
```

1.1.1 Filter for countries and non-projected data

Remove all rows that contain data which is: * not related to a single country (e.g. to whole regions) * contains projected data about the future

```
[8]: countryIndicator=(dataFull["LocTypeName"]=="Country/Area")
timeIndicator=(dataFull["Time"]<=2022)

data=dataFull[countryIndicator & timeIndicator]

print(f"remaining columns: {data.shape[0]}")
```

```
remaining columns: 17301
```

```
[20]: minyear,maxyear=[data["Time"].min(),data["Time"].max()]
print([minyear,maxyear])
```

```
[1950, 2022]
```

```
[9]: # for first orientation, print a list of countries contained in the dataset:
countryList=data["Location"].unique()
```

```
print(countryList)
```

```
['Burundi' 'Comoros' 'Djibouti' 'Eritrea' 'Ethiopia' 'Kenya' 'Madagascar'
 'Malawi' 'Mauritius' 'Mayotte' 'Mozambique' 'Réunion' 'Rwanda'
 'Seychelles' 'Somalia' 'South Sudan' 'Uganda'
 'United Republic of Tanzania' 'Zambia' 'Zimbabwe' 'Angola' 'Cameroon'
 'Central African Republic' 'Chad' 'Congo'
 'Democratic Republic of the Congo' 'Equatorial Guinea' 'Gabon'
 'Sao Tome and Principe' 'Algeria' 'Egypt' 'Libya' 'Morocco' 'Sudan'
 'Tunisia' 'Western Sahara' 'Botswana' 'Eswatini' 'Lesotho' 'Namibia'
 'South Africa' 'Benin' 'Burkina Faso' 'Cabo Verde' "Côte d'Ivoire"
 'Gambia' 'Ghana' 'Guinea' 'Guinea-Bissau' 'Liberia' 'Mali' 'Mauritania'
 'Niger' 'Nigeria' 'Saint Helena' 'Senegal' 'Sierra Leone' 'Togo'
 'Kazakhstan' 'Kyrgyzstan' 'Tajikistan' 'Turkmenistan' 'Uzbekistan'
 'China' 'China, Hong Kong SAR' 'China, Macao SAR'
 'China, Taiwan Province of China' "Dem. People's Republic of Korea"
 'Japan' 'Mongolia' 'Republic of Korea' 'Afghanistan' 'Bangladesh'
 'Bhutan' 'India' 'Iran (Islamic Republic of)' 'Maldives' 'Nepal'
 'Pakistan' 'Sri Lanka' 'Brunei Darussalam' 'Cambodia' 'Indonesia'
 "Lao People's Democratic Republic" 'Malaysia' 'Myanmar' 'Philippines'
 'Singapore' 'Thailand' 'Timor-Leste' 'Viet Nam' 'Armenia' 'Azerbaijan'
 'Bahrain' 'Cyprus' 'Georgia' 'Iraq' 'Israel' 'Jordan' 'Kuwait' 'Lebanon'
 'Oman' 'Qatar' 'Saudi Arabia' 'State of Palestine' 'Syrian Arab Republic'
 'Türkiye' 'United Arab Emirates' 'Yemen' 'Belarus' 'Bulgaria' 'Czechia'
 'Hungary' 'Poland' 'Republic of Moldova' 'Romania' 'Russian Federation'
 'Slovakia' 'Ukraine' 'Denmark' 'Estonia' 'Faroe Islands' 'Finland'
 'Guernsey' 'Iceland' 'Ireland' 'Isle of Man' 'Jersey' 'Latvia'
 'Lithuania' 'Norway' 'Sweden' 'United Kingdom' 'Albania' 'Andorra'
 'Bosnia and Herzegovina' 'Croatia' 'Gibraltar' 'Greece' 'Holy See'
 'Italy' 'Kosovo (under UNSC res. 1244)' 'Malta' 'Montenegro'
 'North Macedonia' 'Portugal' 'San Marino' 'Serbia' 'Slovenia' 'Spain'
 'Austria' 'Belgium' 'France' 'Germany' 'Liechtenstein' 'Luxembourg'
 'Monaco' 'Netherlands' 'Switzerland' 'Anguilla' 'Antigua and Barbuda'
 'Aruba' 'Bahamas' 'Barbados' 'Bonaire, Sint Eustatius and Saba'
 'British Virgin Islands' 'Cayman Islands' 'Cuba' 'Curaçao' 'Dominica'
 'Dominican Republic' 'Grenada' 'Guadeloupe' 'Haiti' 'Jamaica'
 'Martinique' 'Montserrat' 'Puerto Rico' 'Saint Barthélemy'
 'Saint Kitts and Nevis' 'Saint Lucia' 'Saint Martin (French part)'
 'Saint Vincent and the Grenadines' 'Sint Maarten (Dutch part)'
 'Trinidad and Tobago' 'Turks and Caicos Islands'
 'United States Virgin Islands' 'Belize' 'Costa Rica' 'El Salvador'
 'Guatemala' 'Honduras' 'Mexico' 'Nicaragua' 'Panama' 'Argentina'
 'Bolivia (Plurinational State of)' 'Brazil' 'Chile' 'Colombia' 'Ecuador'
 'Falkland Islands (Malvinas)' 'French Guiana' 'Guyana' 'Paraguay' 'Peru'
 'Suriname' 'Uruguay' 'Venezuela (Bolivarian Republic of)' 'Bermuda'
 'Canada' 'Greenland' 'Saint Pierre and Miquelon'
 'United States of America' 'Australia' 'New Zealand' 'Fiji'
 'New Caledonia' 'Papua New Guinea' 'Solomon Islands' 'Vanuatu' 'Guam'
 'Kiribati' 'Marshall Islands' 'Micronesia (Fed. States of)' 'Nauru'
 'Northern Mariana Islands' 'Palau' 'American Samoa' 'Cook Islands'
 'French Polynesia' 'Niue' 'Samoa' 'Tokelau' 'Tonga' 'Tuvalu'
 'Wallis and Futuna Islands']
```

1.1.2 Shorten some country names for more compact labelling

```
[10]: substitutions={"United States of America":"USA"}
```

```
[11]: for long,short in substitutions.items():
          data.loc[data["Location"]==long,"Location"]=short
```

1.1.3 Load and merge with continent data

Now load load a separate CSV file which contains for each country the associated continent. It has been extracted from the metadata provided with the original dataset at:

https://population.un.org/wpp/Download/Documentation/Documentation/

Specifically, from this file:

https://population.un.org/wpp/Download/Files/4_Metadata/WPP2022_F01_LOCATIONS.XLSX

Shortened some continent names.

```
[12]: dataContinents=pd.read_csv("data/UN_population/continent_association.csv",\
              dtype={"SDMX_code":str,"Continent":str})
```

```
[13]: dataContinents.dtypes
```

```
[13]: SDMX_code    object
      Continent    object
      dtype: object
```

```
[14]: Continents=list(dataContinents["Continent"].unique())
      print(Continents)
```

```
['Africa', 'Asia', 'Europe', 'Central and South America', 'North America',
'Oceania']
```

```
[15]: # merge with continent data
      data=pd.merge(data,dataContinents,on="SDMX_code")
```

```
[17]: # this will later allow us to assign colors according to continents
      continentColors={a:colors[i] for i,a in enumerate(Continents)}
```

1.2 Visualizations

1.2.1 Key indicators of some selected countries over years

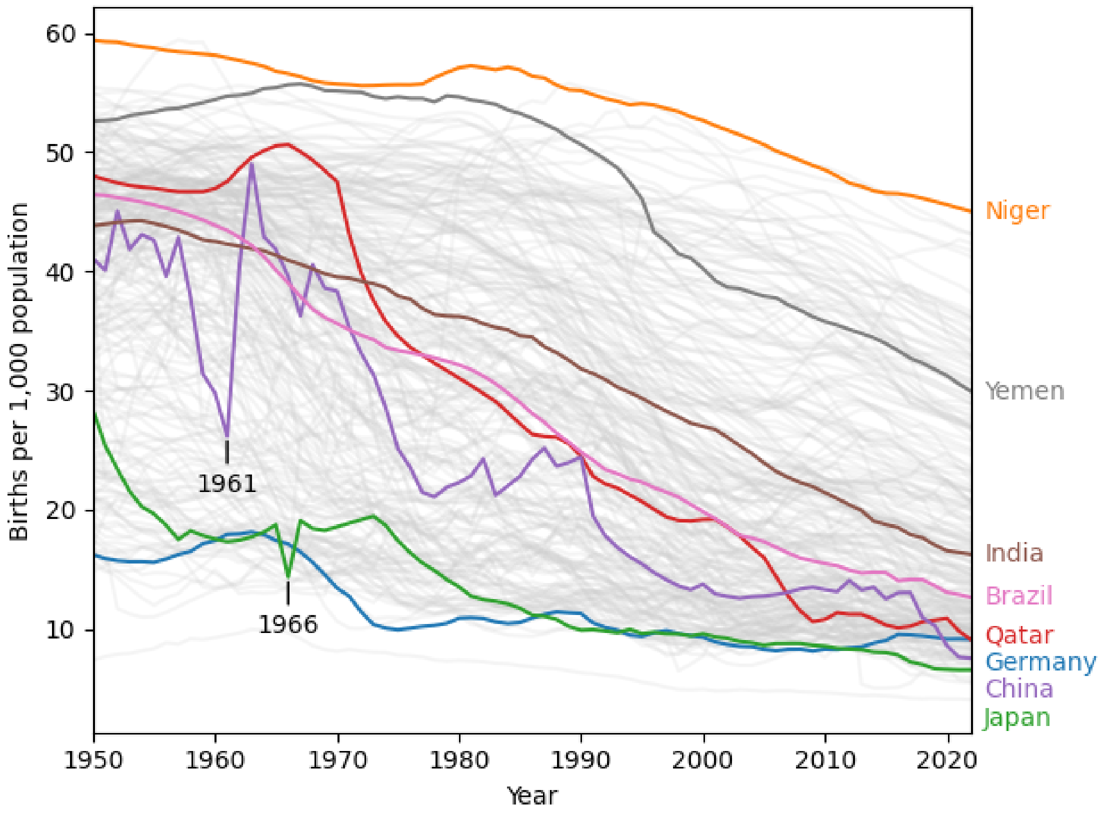
The following plots show: * birth rate * child mortality * median age

Each for a (relevant) selection of countries over time.

Birth rate

[21]:
```
countrySelList=["Germany","Niger","Japan","Qatar",\
        "China","India","Brazil","Yemen"]

fig=plt.figure()
ax=fig.add_subplot()

for country in countryList:
    if country not in countrySelList:
        dataCountry=data[data["Location"]==country]
        ax.plot(dataCountry["Time"],dataCountry["CBR"],c=cm.gray(0.8),alpha=0.
↪2,zorder=-1)


for country in countrySelList:
    dataCountry=data[data["Location"]==country]
    ax.plot(dataCountry["Time"],dataCountry["CBR"],label=country)


# directl labelling
offsets={\
        "Germany":-10,"Niger":0,"Japan":-20,"Qatar":2,\
        "China":-13,"India":0,"Brazil":0,"Yemen":0\
        }
for i,country in enumerate(countrySelList):
    y=data.loc[(data["Location"]==country)&(data["Time"]==maxyear),"CBR"].
↪iloc[0]
    ax.
↪annotate(text=country,xy=(maxyear,y),va="center",ha="left",textcoords="offset␣
↪points",\
            xytext=(5,offsets[country]),c=colors[i],\
            arrowprops=None)

# fire horse annotation
y=data.loc[(data["Location"]=="Japan")&(data["Time"]==1966),"CBR"].iloc[0]
ax.annotate(text="1966",xy=(1966,y),va="center",ha="center",textcoords="offset␣
↪points",\
            xytext=(0,-20),c="k",\
            arrowprops={"arrowstyle":"-"})
```

```
# great leap
y=data.loc[(data["Location"]=="China")&(data["Time"]==1961),"CBR"].iloc[0]
ax.annotate(text="1961",xy=(1961,y),va="center",ha="center",textcoords="offset␣
 ↪points",\
            xytext=(0,-20),c="k",\
            arrowprops={"arrowstyle":"-"})

plt.xlabel("Year")
plt.ylabel("Births per 1,000 population")
ax.set_xlim([minyear,maxyear])
plt.tight_layout()
plt.show()
```

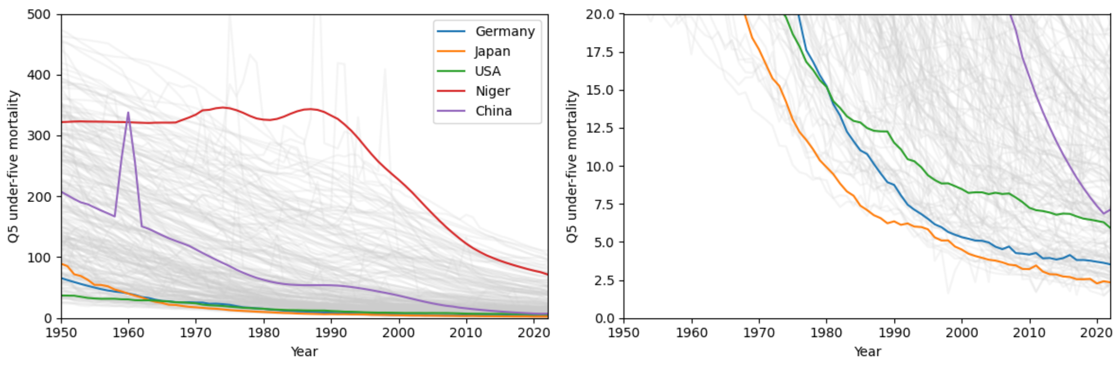


Annotations: * 1966, Japan: "Fire horse superstition" leads to a lowered birth rate in that year * 1961, China: Birth rate dropped significantly during the "Great Leap Forward" from 1958 to 1962.

Q5 Child mortality rate

- Under-five Mortality Rate (deaths under age 5 per 1,000 live births)

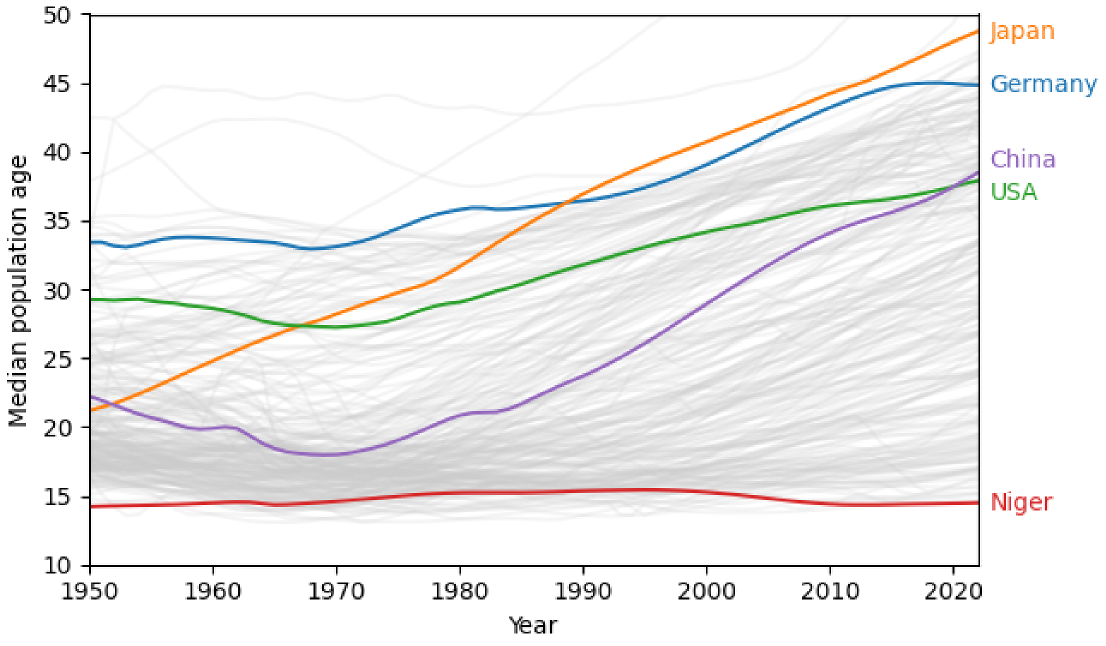
[22]:
```
countrySel=["Germany","Japan","USA","Niger","China"]

fig=plt.figure(figsize=(12,4))
ax=fig.add_subplot(1,2,1)
for country in countryList:
    if country not in countrySel:
        dataCountry=data[data["Location"]==country]
        plt.plot(dataCountry["Time"],dataCountry["Q5"],c=cm.gray(0.8),alpha=0.
 ↪2,zorder=-1)
for i,country in enumerate(countrySel):
    dataCountry=data[data["Location"]==country]
    plt.plot(dataCountry["Time"],dataCountry["Q5"],c=colors[i],label=country)
plt.xlim([1950,2022])
plt.ylim([0,500])
plt.legend()
plt.xlabel("Year")
plt.ylabel("Q5 under-five mortality")

ax=fig.add_subplot(1,2,2)
for country in countryList:
    if country not in countrySel:
        dataCountry=data[data["Location"]==country]
        plt.plot(dataCountry["Time"],dataCountry["Q5"],c=cm.gray(0.8),alpha=0.
 ↪2,zorder=-1)
for i,country in enumerate(countrySel):
    dataCountry=data[data["Location"]==country]
    plt.plot(dataCountry["Time"],dataCountry["Q5"],c=colors[i])
plt.xlim([1950,2022])
plt.ylim([0,20])

plt.xlabel("Year")
plt.ylabel("Q5 under-five mortality")

plt.tight_layout()
plt.show()
```



- In China, the ”Great Leap Forward” lead to a strongly increased child-mortality around 1960. Visualization discussion:
- Did not go for direct labelling here for several reasons: values all really close at end; no space between sub-figures; fewer distinct countries, so also less confusing.

Median population age

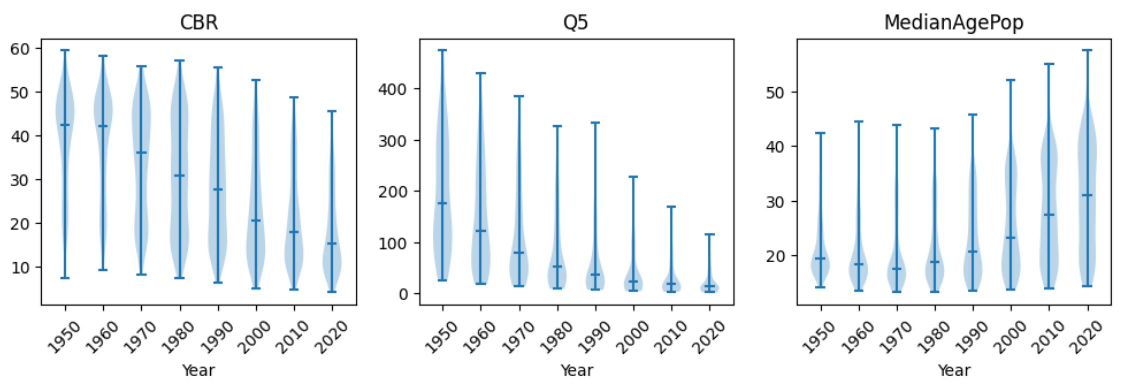
[23]:
```python
countrySelList=["Germany","Japan","USA","Niger","China"]

fig=plt.figure(figsize=(12,4))
ax=fig.add_subplot(1,2,1)

for country in countryList:
    if country not in countrySelList:
        dataCountry=data[data["Location"]==country]
        ax.plot(dataCountry["Time"],dataCountry["MedianAgePop"],c=cm.gray(0.
 ↪8),alpha=0.2,zorder=-1)


for i,country in enumerate(countrySelList):
    dataCountry=data[data["Location"]==country]
    plt.plot(dataCountry["Time"],dataCountry["MedianAgePop"],c=colors[i])

# directl labelling
offsets={\
        "Germany":0,"Japan":0,"USA":-5,"Niger":0,"China":+5
        }
for i,country in enumerate(countrySelList):
    y=data.
 ↪loc[(data["Location"]==country)&(data["Time"]==maxyear),"MedianAgePop"].
 ↪iloc[0]
    ax.
 ↪annotate(text=country,xy=(maxyear,y),va="center",ha="left",textcoords="offset␣
 ↪points",\
            xytext=(5,offsets[country]),c=colors[i],\
            arrowprops=None)


plt.ylim([10,50])
plt.xlabel("Year")
plt.ylabel("Median population age")
ax.set_xlim([minyear,maxyear])
plt.tight_layout()

plt.show()
```



1.2.2 Evolution of distribution over countries

- Visualize same three quantities
- Instead of highlighting individual countries, show distribution (and its evolution over time) by violin plots.

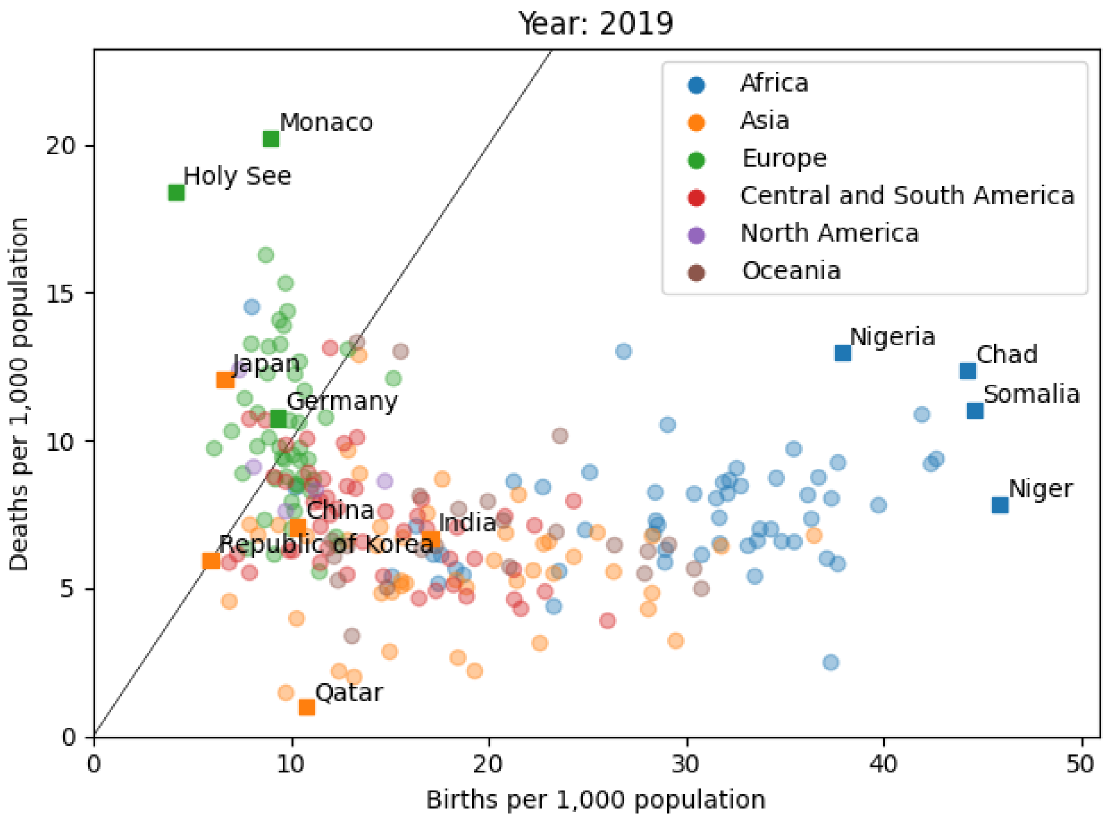
[24]:
```
yearsSel=range(1950,2022,10)
nYearsSel=len(yearsSel)
dataG=data.groupby("Time")
keyList=["CBR","Q5","MedianAgePop"]
nKeys=len(keyList)
fig=plt.figure(figsize=(4*nKeys,3))
for i,key in enumerate(keyList):
    ax=fig.add_subplot(1,nKeys,i+1)
    ptdata=[]
    for year,dat in dataG:
        if year in yearsSel:
            ptdata.append(dat[key])
    #plt.boxplot(ptdata,labels=yearsSel)
    ax.violinplot(ptdata,showmedians=True)
    ax.set_xticks(range(1,nYearsSel+1),yearsSel,rotation=45)
    plt.title(key)
    plt.xlabel("Year")
plt.show()
```



1.3 Birth and death rates in a given year for all countries

[25]:
```python
# highlight these countries by direct labelling
countrySelList=["Germany","Chad","Somalia","Niger","Monaco","Holy See","Japan",\
        "Qatar","Nigeria","Republic of Korea","China","India"]
```

[26]:
```python
year=2019
dataYear=data[data["Time"]==year]
sel=dataYear["Location"].isin(countrySelList)
for s in [False,True]:
    plt.scatter(dataYear.loc[sel==s,"CBR"],dataYear.loc[sel==s,"CDR"],\
            alpha={False:0.4,True:1.}[s],marker={False:"o",True:"s"}[s],\
            c=dataYear.loc[sel==s,"Continent"].map(continentColors))

ax=plt.gca()
for country in countrySelList:
    x,y=np.array(dataYear.loc[dataYear["Location"]==country,["CBR","CDR"]])[0]
    ax.annotate(country,(x,y),xycoords="data",xytext=(3,3),textcoords="offset␣
 ↪points",\
    arrowprops=None)

# add continent legend
for i,cont in enumerate(Continents):
    plt.scatter([],[],marker="o",color=continentColors[cont],label=cont)

# ax aspect
#ax.set_aspect(1.)
# manual axes limits
CBRmax,CDRmax=dataYear[["CBR","CDR"]].max()
CBRmax+=5
CDRmax+=3
ax.set_xlim([0,CBRmax])
ax.set_ylim([0,CDRmax])
# "balance line"
ax.plot([0,CDRmax],[0,CDRmax],zorder=-1,lw=0.5,c="k")
plt.legend(loc="upper right")
plt.title(f"Year: {year}")
plt.xlabel("Births per 1,000 population")
plt.ylabel("Deaths per 1,000 population")
plt.tight_layout()
plt.show()
```

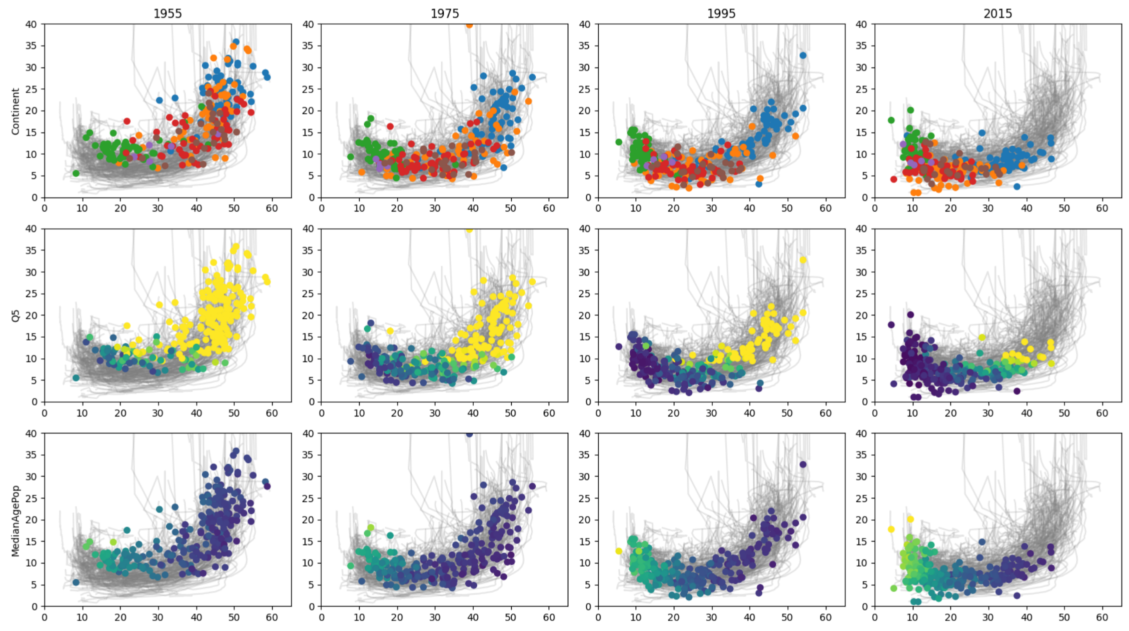


Visualize evolution over time

- Use small multiples
- First, as above use color to encode continent
- Then use color to encode under-five mortality and median age, as above.

[27]:
```
yearsSel=list(range(2015,1950,-20))[::-1]
#print(yearsSel)
nYearsSel=len(yearsSel)
keyList=["Continent","Q5","MedianAgePop"]
```

```
nRows=len(keyList)

colorFunction={"Continent":lambda x : x.map(continentColors),\
              "Q5": lambda x : cm.viridis(np.clip(x,0,100)/100),\
              "MedianAgePop": lambda x : cm.viridis((np.clip(x,10,50)-10)/40)\
              }

# background: lines of each country over time
fig=plt.figure(figsize=(4*nYearsSel,3*nRows))
for j,key in enumerate(keyList):
    for i,year in enumerate(yearsSel):
        ax=fig.add_subplot(nRows,nYearsSel,j*nYearsSel+i+1)
        for country in countryList:
            dataSel=data[data["Location"]==country]
            ax.plot(dataSel["CBR"],dataSel["CDR"],c=cm.gray(0.5),alpha=0.
 ↪2,zorder=-1)
        dataYear=data[data["Time"]==year]
        ax.
 ↪scatter(dataYear["CBR"],dataYear["CDR"],c=colorFunction[key](dataYear[key]))
        ax.set_ylim([0,40])
        ax.set_xlim([0,65])
        if j==0:
            plt.title(year)
        if i==0:
            plt.ylabel(key)
plt.tight_layout()
plt.show()
```



Color scales:

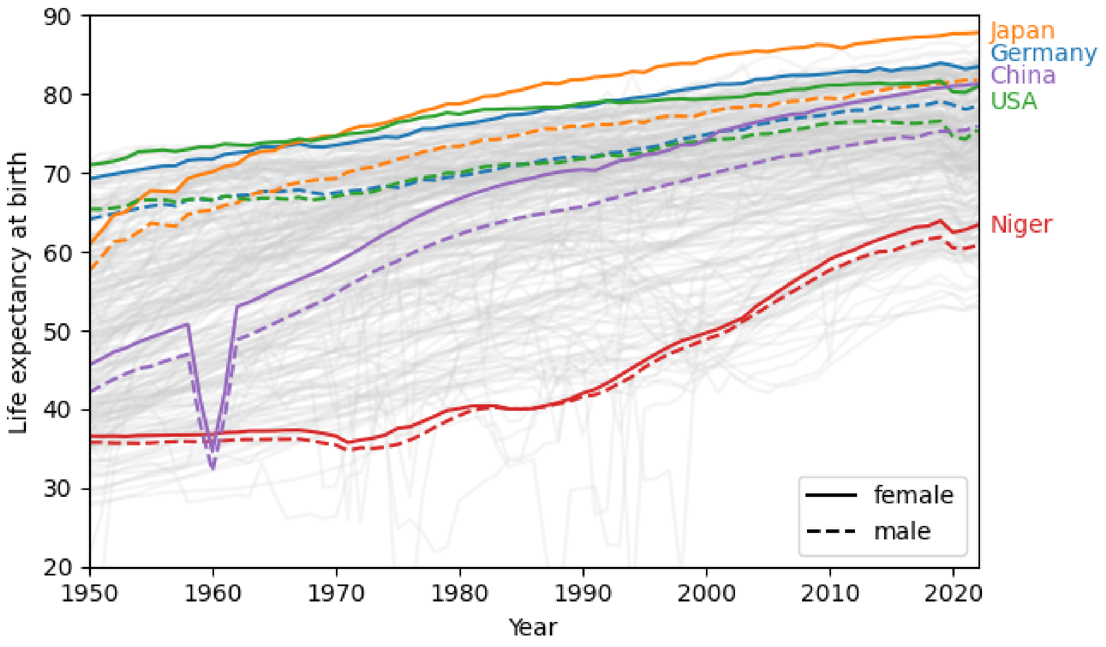
- Continents as above
- under-five mortality: 0 to 100+
- median age: 10- to 50+ #### Take aways:
- general trend towards lower birth rates, this is probably causally related to child mortality (but of course, we cannot know for sure based on this data)
- as birth rates lower, median age rises, and thus eventually so does the death rate

1.3.1 Life expectancy

Simple overview First, a simple overview. Evolution of life expectancy of women and men of some selected countries over time.

[28]:
```
fig=plt.figure(figsize=(12,4))
ax=fig.add_subplot(1,2,1)

countrySelList=["Germany","Japan","USA","Niger","China"]

for country in countryList:
    if country not in countrySelList:
        dataCountry=data[data["Location"]==country]
        ax.plot(dataCountry["Time"],dataCountry["LEx"],c=cm.gray(0.8),alpha=0.
↪2,zorder=-1)


for i,country in enumerate(countrySelList):
    dataCountry=data[data["Location"]==country]
    plt.plot(dataCountry["Time"],dataCountry["LExMale"],c=colors[i],ls="dashed")
    plt.
↪plot(dataCountry["Time"],dataCountry["LExFemale"],c=colors[i])#,ls="dotted")


# directl labelling
offsets={\
        "Germany":+5,"Japan":0,"USA":-7,"Niger":0,"China":+3
        }
for i,country in enumerate(countrySelList):
    y=data.loc[(data["Location"]==country)&(data["Time"]==maxyear),"LExFemale"].
↪iloc[0]
    ax.
↪annotate(text=country,xy=(maxyear,y),va="center",ha="left",textcoords="offset␣
↪points",\
            xytext=(5,offsets[country]),c=colors[i],\
            arrowprops=None)
```

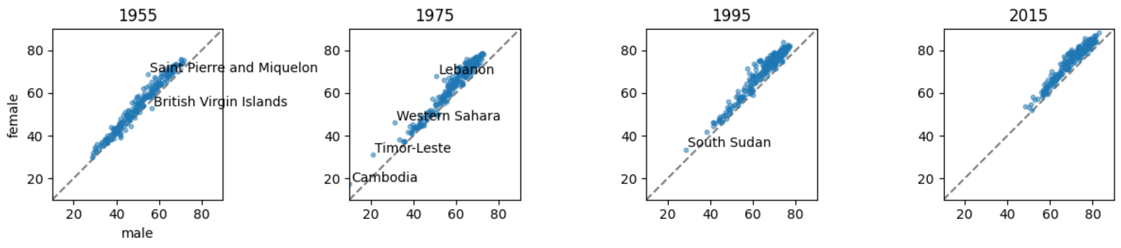
```
plt.plot([],[],ls="solid",c="k",label="female")
plt.plot([],[],ls="dashed",c="k",label="male")
plt.legend(loc="lower right")
plt.ylim([20,90])
plt.xlabel("Year")
plt.ylabel("Life expectancy at birth")
ax.set_xlim([minyear,maxyear])
plt.tight_layout()

plt.show()
```



Background shows average life expectancy at birth of all countries for context.

Male vs female scatter plot

[29]:
```
yearsSel=list(range(2015,1950,-20))[::-1]
nYearsSel=len(yearsSel)

highlights={1955:["British Virgin Islands","Saint Pierre and Miquelon"],\
        1975:["Cambodia","Lebanon","Western Sahara","Timor-Leste"],\
        1995:["South Sudan"],2015:[]}

fig=plt.figure(figsize=(nYearsSel*3,3))
for i,year in enumerate(yearsSel):
    ax=fig.add_subplot(1,nYearsSel,i+1,aspect=1.)
    dataYear=data[data["Time"]==year]
    plt.
  ↪scatter(dataYear["LExMale"],dataYear["LExFemale"],s=10,marker="o",alpha=0.5)
    plt.xlim([10,90])
    plt.ylim([10,90])
    plt.plot([10,90],[10,90],c=cm.gray(0.5),ls="dashed",zorder=-1)

    # annotations
    for country in highlights[year]:
        x=dataYear.loc[(dataYear["Location"]==country),"LExMale"].iloc[0]
        y=dataYear.loc[(dataYear["Location"]==country),"LExFemale"].iloc[0]
        plt.annotate(country,xy=(x+1,y+1))

    plt.title(year)
    if i==0:
        plt.ylabel("female")
        plt.xlabel("male")
plt.tight_layout()
plt.show()
```

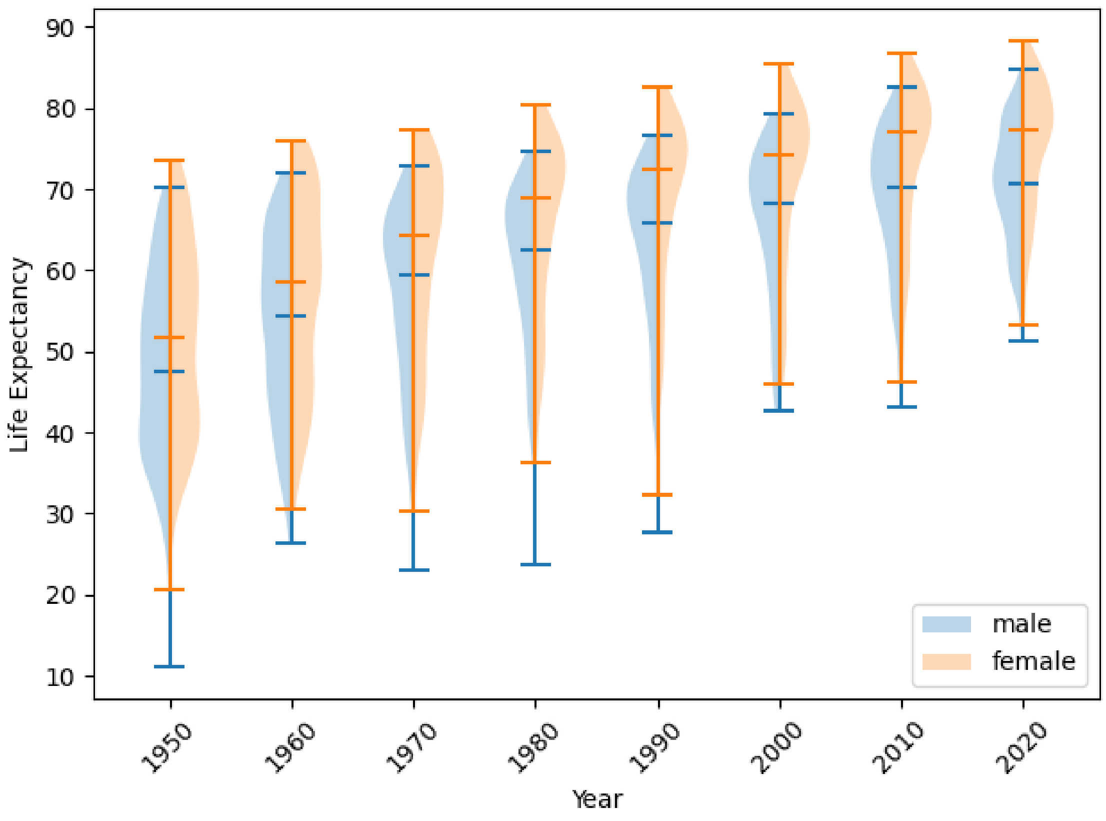


Half-violin plot

- show male and female life expectancy for various years as half-violin plot
- allows for decent statistical overview, and finer temporal resolution
- does not immediately allow to see correlation between the two, at level of individual countries

[30]:
```
# code based on:
# https://stackoverflow.com/questions/29776114/half-violin-plot-in-matplotlib
```

[31]:
```
yearsSel=range(1950,2022,10)
nYearsSel=len(yearsSel)
dataG=data.groupby("Time")
fig=plt.figure()
ax=fig.add_subplot()
ptdatam=[]
ptdataf=[]
for year,dat in dataG:
    if year in yearsSel:
        ptdatam.append(dat["LExMale"])
        ptdataf.append(dat["LExFemale"])
v1=ax.violinplot(ptdatam,showmeans=False, showextrema=True, showmedians=True)
for b in v1['bodies']:
    m = np.mean(b.get_paths()[0].vertices[:, 0])
    b.get_paths()[0].vertices[:, 0] = np.clip(b.get_paths()[0].vertices[:, 0],␣
  ↪-np.inf, m)

v2=ax.violinplot(ptdataf,showmeans=False, showextrema=True, showmedians=True)
for b in v2['bodies']:
    m = np.mean(b.get_paths()[0].vertices[:, 0])
    b.get_paths()[0].vertices[:, 0] = np.clip(b.get_paths()[0].vertices[:, 0],␣
  ↪m, np.inf)


ax.set_xticks(range(1,nYearsSel+1),yearsSel,rotation=45)
plt.ylabel("Life Expectancy")
plt.xlabel("Year")
plt.legend([v1['bodies'][0],v2['bodies'][0]],['male', 'female'],loc="lower␣
  ↪right")
plt.tight_layout()
plt.show()
```



1.4 More to explore

- In principle, this dataset allows for even more exploration, e.g.~of migration data.

1.5 Some final comments

- Plot types: scatter and line plots, a few violin plots; everything static in time; no interaction
- Not a single map in this notebook
- Used position and color for encoding quantities, this got us quite far
- Not a prerequisite for a good project to use all sorts of fancy plot types, more important: use them in a good way.
- But of course: feel free to be creative and explore/experiment with other plot types (e.g. networks), animations, interactive plots. But do not create ducks :)

```
[ ]:
```