2023-04-17_WorldDemographics-Test-01

April 18, 2023

[4]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.cm as cm

%matplotlib inline
```

1 Naive first 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)

1.1 Import data

[5]:
```
# 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
```

[6]:
```
dataFull=pd.read_csv("data/WPP2022_Demographic_Indicators_Medium.
 ↪csv",sep=",",dtype=dtypeDict)
```

[7]:
```
dataFull.keys()
```

[7]:
```
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')
```

[8]:
```
dataFull.shape
```

[8]:
```
(43472, 67)
```

1.1.1 Filter for countries and non-projected data

[9]:
```
countryIndicator=(dataFull["LocTypeName"]=="Country/Area")
timeIndicator=(dataFull["Time"]<=2022)

data=dataFull[countryIndicator & timeIndicator]

data.shape
```

[9]:
```
(17301, 67)
```

[10]:
```
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'
```

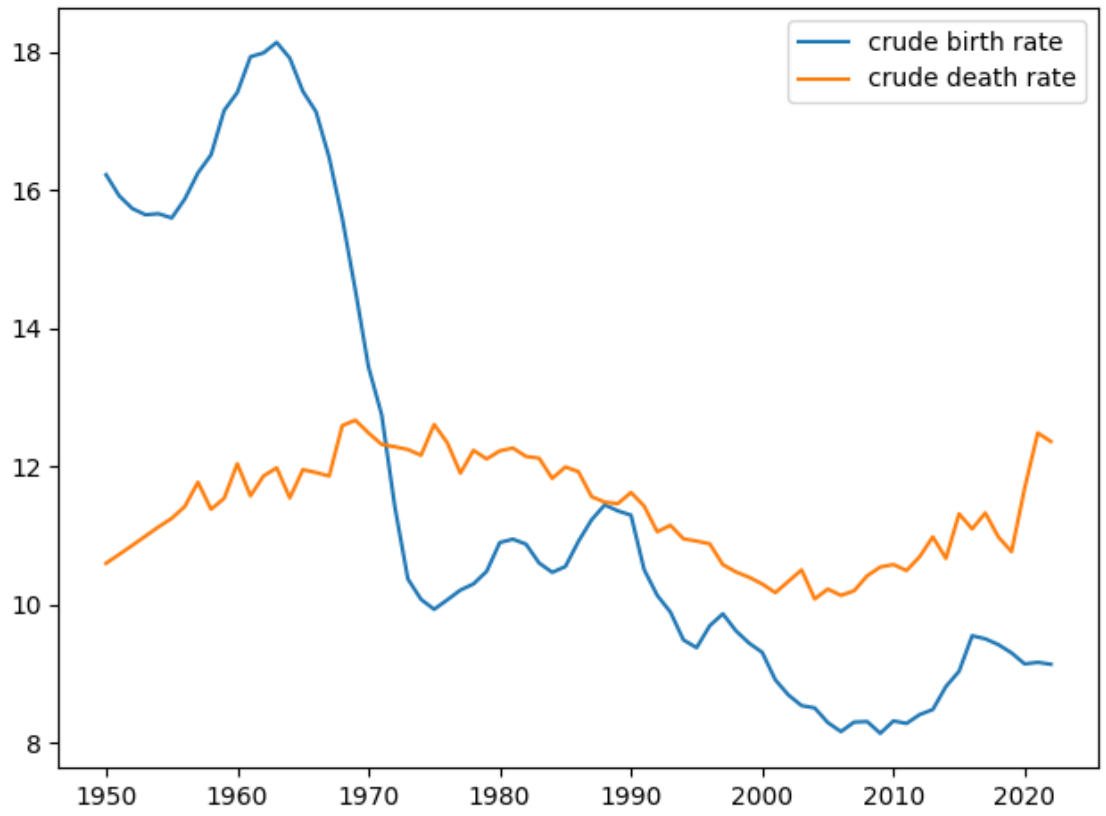
```
'Oman' 'Qatar' 'Saudi Arabia' 'State of Palestine' 'Syrian Arab Republic'
'Türkiye' 'United Arab Emirates' 'Yemen' 'Belarus' 'Bulgaria' 'Czechia'
'Hungary' 'Poland' 'Republic of Moldova' 'Romania' 'Russian Federation'
'Slovakia' 'Ukraine' 'Denmark' 'Estonia' 'Faroe Islands' 'Finland'
'Guernsey' 'Iceland' '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']
```

1.2 Exploring dataset

1.2.1 First naive test: birth and death rates in Germany

[11]:
```
# CBR: "crude birth rate": births per 1000 persons in that year
# CDR: "crude death rate"
```

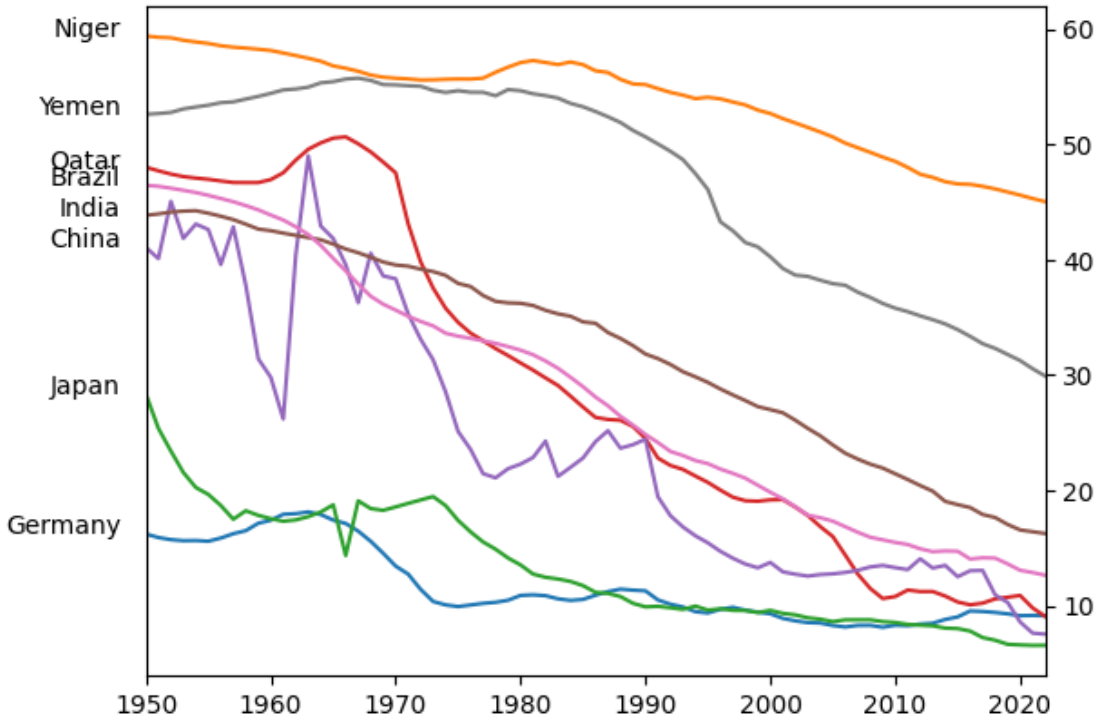
[12]:
```
indicatorCountry=(data["Location"]=="Germany")
dataCountry=data[indicatorCountry]
fig=plt.figure()
plt.plot(dataCountry["Time"],dataCountry["CBR"],label="crude birth rate")
plt.plot(dataCountry["Time"],dataCountry["CDR"],label="crude death rate")
plt.legend()
plt.tight_layout()
plt.show()
```



1.2.2 Birth rates some selected countries over years

[14]:
```
countrySelList=["Germany","Niger","Japan","Qatar",\
        "China","India","Brazil","Yemen"]

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

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

minyear=data["Time"].min()
maxyear=data["Time"].max()
ax.set_xlim([minyear,maxyear])


# explicit legend
#plt.legend()
```

```
# directl labelling
if True:
    ax.yaxis.tick_right()
    for country in countrySelList:
        y=data.loc[(data["Location"]==country)&(data["Time"]==minyear),"CBR"].
 ↪iloc[0]
        ax.text(x=minyear-2,y=y,s=country,horizontalalignment="right")

plt.show()
```

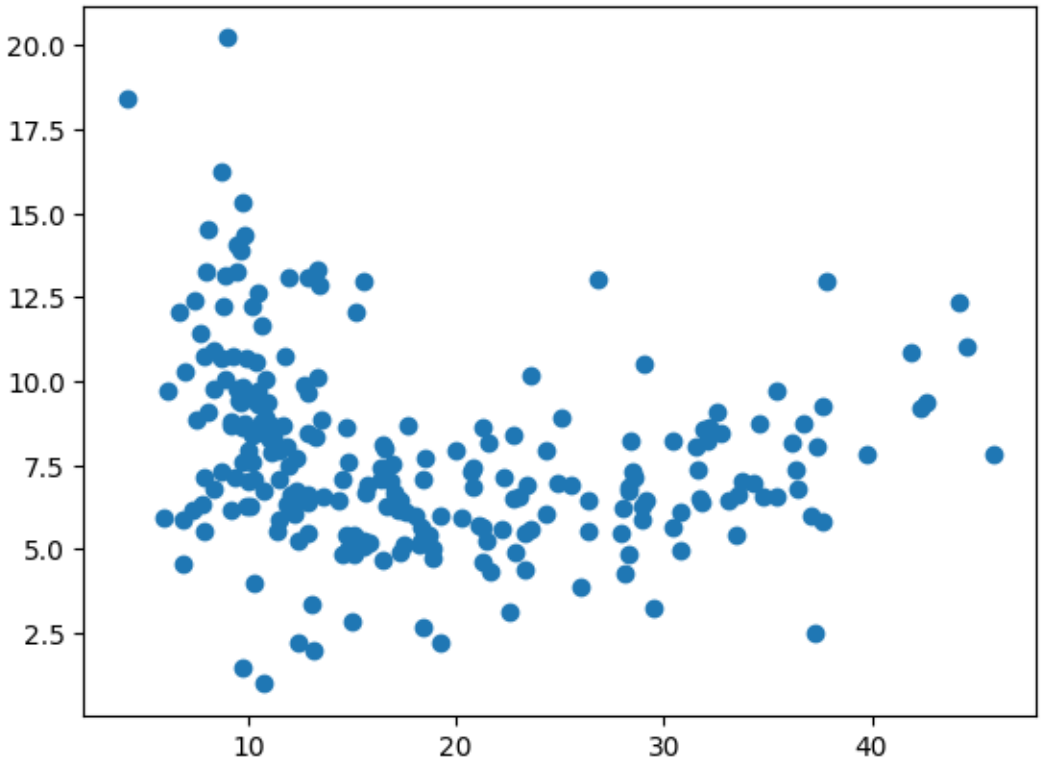


[15]:
```
# could now proceed to annotate / discuss peculiar artefacts in data:
# in japan 1966: "fire horse superstition"
# in china: great leap forward and one child policy
```

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

[17]:
```
year=2019
dataYear=data[(data["Time"]==year)]
```

[18]:
```
plt.scatter(dataYear["CBR"],dataYear["CDR"])
plt.show()
```



[19]:
```
# now: we don't know which country is which; too many for legend
# try hover technique
```

[20]:
```
import plotly.express as px
```

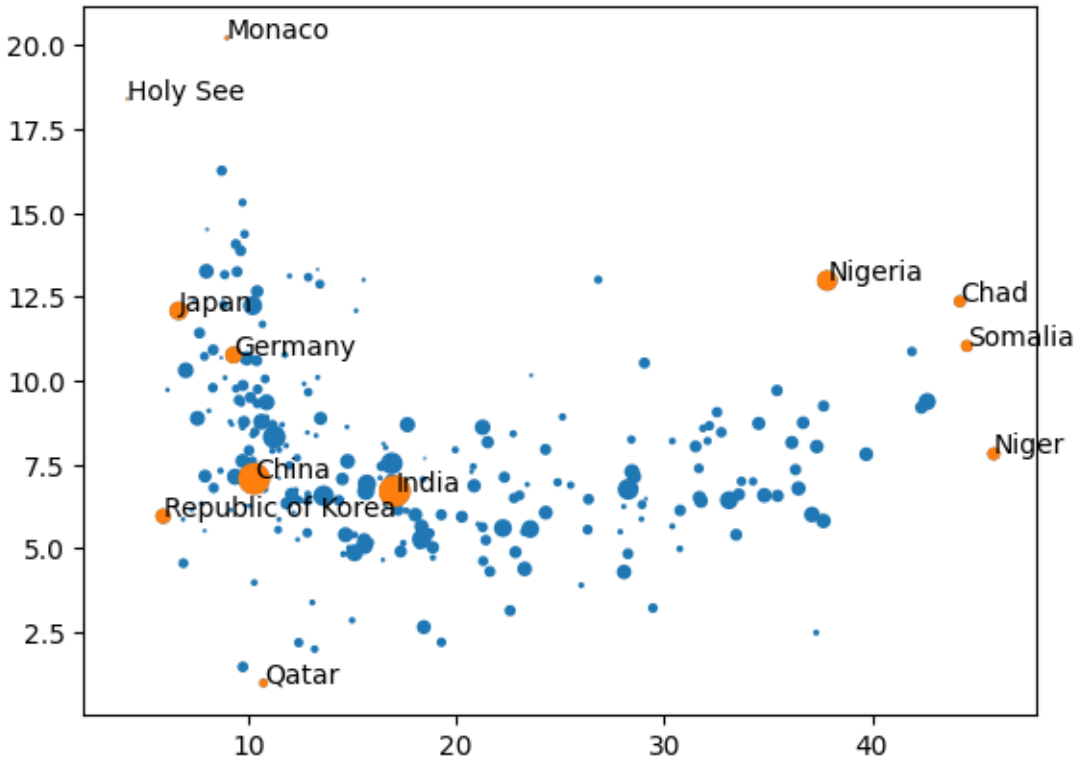
[26]:
```
fig = px.scatter(dataYear, x="CBR", y="CDR", hover_data=['Location'],\
        #size="TPopulation1Jan"\
        #color="Under5Deaths"\
        )
fig.show()
```

[36]:
```
# also experiment: map other columns to additional attributes

# but this is difficult for getting a "single glance impression"
# instead: static plot with some directly labelled "interesting" countries
```

[22]:
```
countrySelList=["Germany","Chad","Somalia","Niger","Monaco","Holy See","Japan",\
        "Qatar","Nigeria","Republic of Korea","China","India"]
```

[23]:
```
sList=1E-1*(np.array(dataYear["TPopulation1Jan"])**0.5)
sListSel=1.01E-1*(np.array(dataYear.loc[dataYear["Location"].
 ↪isin(countrySelList),"TPopulation1Jan"])**0.5)
```

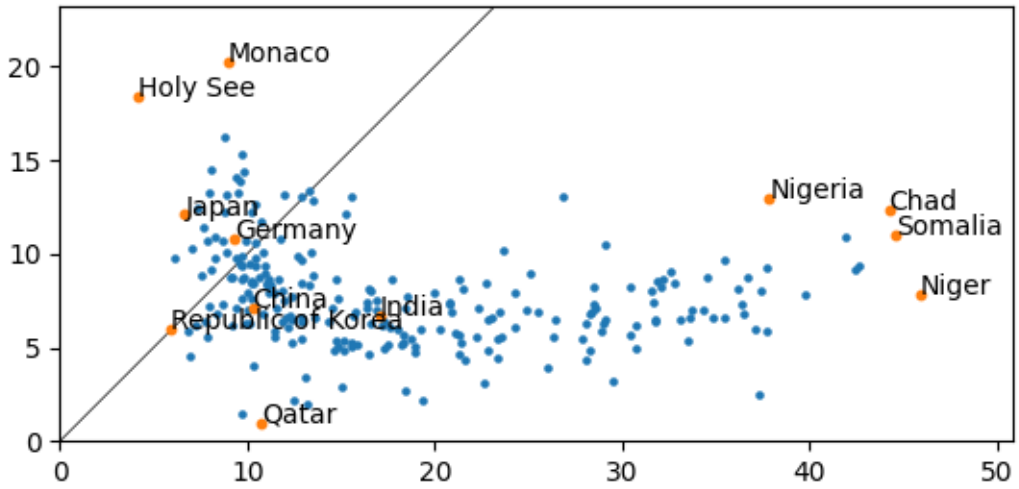
```
plt.scatter(dataYear["CBR"],dataYear["CDR"],\
        s=sList)
plt.scatter(dataYear.loc[dataYear["Location"].isin(countrySelList),"CBR"],\
        dataYear.loc[dataYear["Location"].
  ↪isin(countrySelList),"CDR"],s=sListSel)
ax=plt.gca()
for country in countrySelList:
    x,y=np.array(dataYear.loc[dataYear["Location"]==country,["CBR","CDR"]])[0]
    ax.annotate(country,(x,y))
plt.show()
```



[27]:
```
# fine tuning:
# * e.g. positioning of annotation
```

[28]:
```
plt.scatter(dataYear["CBR"],dataYear["CDR"],s=5)
plt.scatter(dataYear.loc[dataYear["Location"].isin(countrySelList),"CBR"],\
        dataYear.loc[dataYear["Location"].isin(countrySelList),"CDR"],\
        s=10)
ax=plt.gca()
for country in countrySelList:
    x,y=np.array(dataYear.loc[dataYear["Location"]==country,["CBR","CDR"]])[0]
```

```
    ax.annotate(country,(x,y))
# 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.show()
```



[30]:
```
# movement over time
from matplotlib.animation import FuncAnimation
matplotlib.rc('animation',html='jshtml')
```

[31]:
```
fig=matplotlib.figure.Figure(figsize=(8,4))
ax=fig.add_subplot(aspect=1.)
pltobj_pts=ax.scatter([],[],s=5)

# manual axes limits
ax.set_xlim([0,60])
ax.set_ylim([0,30])


def update(year):
    dataYear=data[(data["Time"]==year)]
    # now update scatter-plot object with new vertex positions
```

```
    pltobj_pts.set_offsets(np.array(dataYear[["CBR","CDR"]]))

ani = FuncAnimation(fig, update, frames=np.arange(1955, 2022),
        blit=True,interval=2*1000/20)

ani
```

[31]:
```
<matplotlib.animation.FuncAnimation at 0x7f416164c550>
```

[ ]:
```

```