DAT565/DIT407 Assignment 1

Ola Bratt
ola.bratt@gmail.com

Patrick Attimont
patrickattimont@gmail.com

2024-01-16

Problem 1: Dependency Ratio

In Figure 1 the Dependecy Ratio of Sweden from 1860 to 2022 is show. The ratio is calculated by using the data from SCB [1].

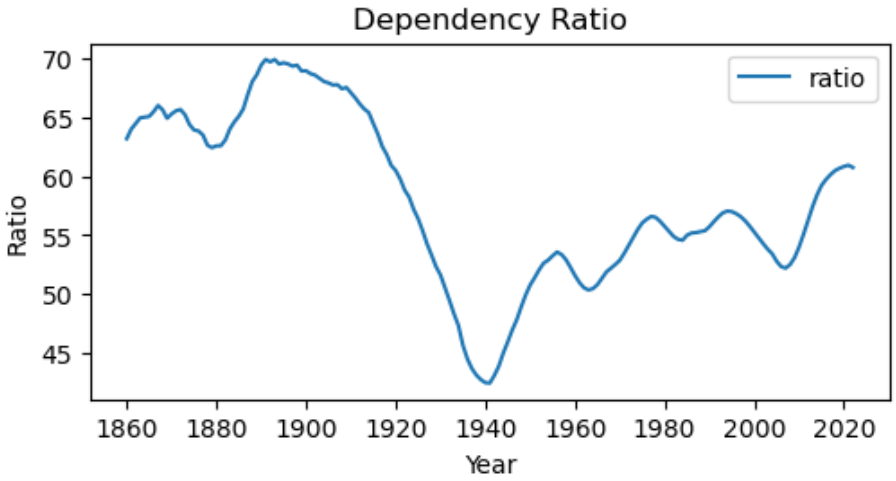

Figure 1: Dependecy ratio

In Figure 2 the diffrent fractions of the population is shown. The fractions are calculated by dividing the population group (children, elderly and total depentent population) with the total population. The fractions are calculated by using the data from SCB [1].

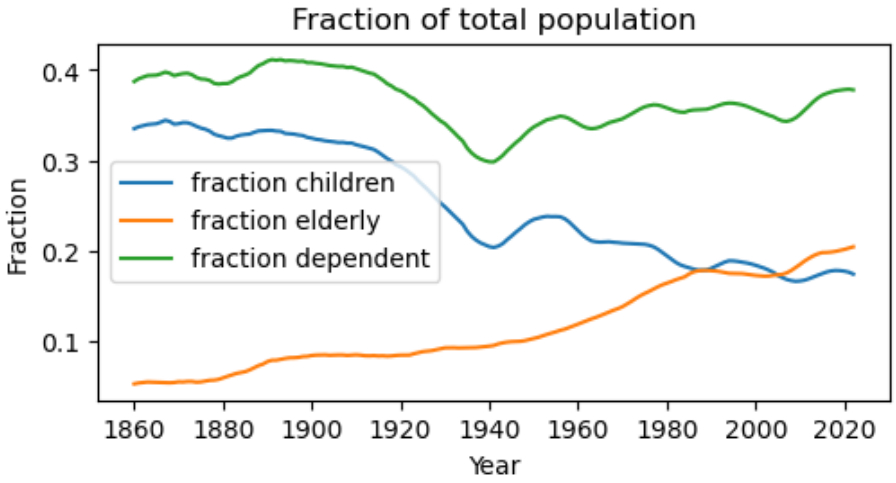

Figure 2: Fractions

Discuss the development of the Swedish population in light of these figures; how have the Swedish demographics changed over the years and why, and relate this to what you know (or can find out) about general trends of population among industrialized countries.

References

[1] Statistiska centralbyrån. *Folkmängden efter ålder och kön. År 1860 - 2022.* Retrieved 2023-10-20. 2023. URL: `https://www.statistikdatabasen.scb.se/pxweb/sv/ssd/START__BE__BE0101__BE0101A/BefolkningR1860N/`.

Appendix: Source Code

```
import numpy as np
import pandas as pd
from matplotlib import pyplot


ratio = lambda children, elderly, labor: 100 * (children + elderly) / labor
total = lambda children, elderly, labor: children + elderly + labor
fraction = lambda part, total: part / total

population = pd.read_csv('swedish_population_by_year_and_sex_1860-2022.csv',
sep=',',)
```

```
# Drop sex column, we don't need it
populationNoSex = population.drop(columns=['sex'])

# Set age to numeric
populationNoSex.at[220,'age'] = 110
populationNoSex.at[221,'age'] = 110
# Convert to numeric
populationNoSex['age'] = pd.to_numeric(populationNoSex['age'], errors='coerce',


# Group by age
classes = populationNoSex.groupby(pd.cut(populationNoSex['age'], [-1, 14, 64, 1
# Drop age column, we don't need it anymore
classes = classes.drop(columns=['age'])
# Transpose
classesT = classes.transpose()
# Apply lambda functions
classesT['ratio'] = classesT.apply(lambda row: ratio(row.iat[0], row.iat[2], ro
classesT['total'] = classesT.apply(lambda row: total(row.iat[0], row.iat[2], ro
classesT['fraction_children'] = classesT.apply(lambda row: fraction(row.iat[0],
classesT['fraction_elderly'] = classesT.apply(lambda row: fraction(row.iat[2],
classesT['fraction_dependent'] = classesT.apply(lambda row: fraction(row.iat[0]

# Convert index to float
years = np.asarray(classesT.index.values, float)

# Plot ratio
fig1, ax1 = pyplot.subplots(figsize=(5, 2.7), layout='constrained')
ax1.plot(years, classesT['ratio'], label='ratio')
ax1.set_xlabel('Year')  # Add an x-label to the axes.
ax1.set_ylabel('Ratio')  # Add a y-label to the axes.
ax1.set_title("Dependency Ratio")  # Add a title to the axes.
ax1.legend()

# Plot fractions
fig2, ax2 = pyplot.subplots(figsize=(5, 2.7), layout='constrained')
ax2.plot(years, classesT['fraction_children'], label='fraction children')
ax2.plot(years, classesT['fraction_elderly'], label='fraction elderly')
ax2.plot(years, classesT['fraction_dependent'], label='fraction dependent')
ax2.set_xlabel('Year')  # Add an x-label to the axes.
ax2.set_ylabel('Fraction')  # Add a y-label to the axes.
ax2.set_title("Fraction of total population")  # Add a title to the axes.
ax2.legend()
```