

DAT565/DIT407 Assignment 1

Ola Bratt
ola.bratt@gmail.com

Patrick Attimont
patrickattimont@gmail.com

2024-01-23

This paper is addressing the assignment 1 study queries within the *Introduction to Data Science & AI* course, DIT407 at the University of Gothenburg and DAT565 at Chalmers. The main source of information for this project is derived from the lectures and Skiena [3]. Assignment 1 focuses on using Python tools, such as Pandas, NumPy, and Matplotlib.

Problem 1: Dependency Ratio

In Figure 1 the Dependency Ratio of Sweden from 1860 to 2022 is shown. The ratio is calculated by using the data from SCB [1]. The ratio is calculated by dividing the total dependent population (children, 0-14 years and elderly, 65+) with the labor force, 15-64 years and multiply with 100 (Equation 1).

$$\text{Dependency ratio} = 100 * \frac{\text{children} + \text{elderly}}{\text{labor force}} \quad (1)$$

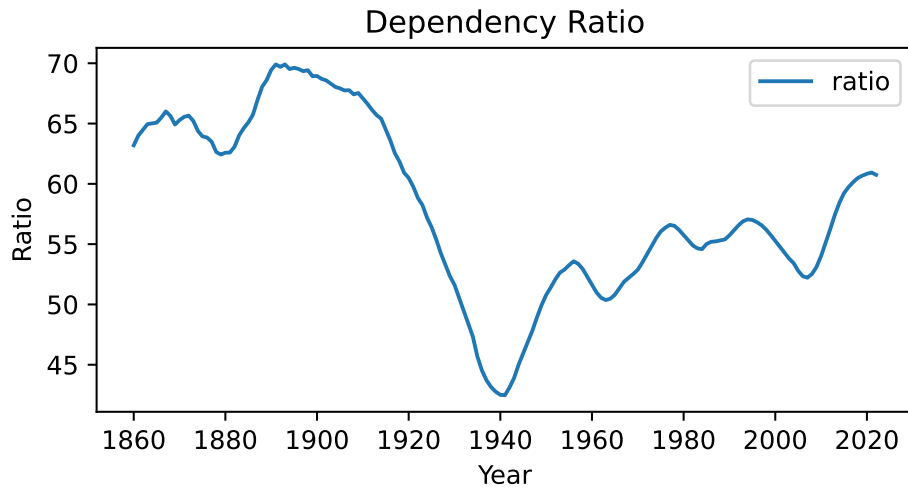


Figure 1: Dependency ratio

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

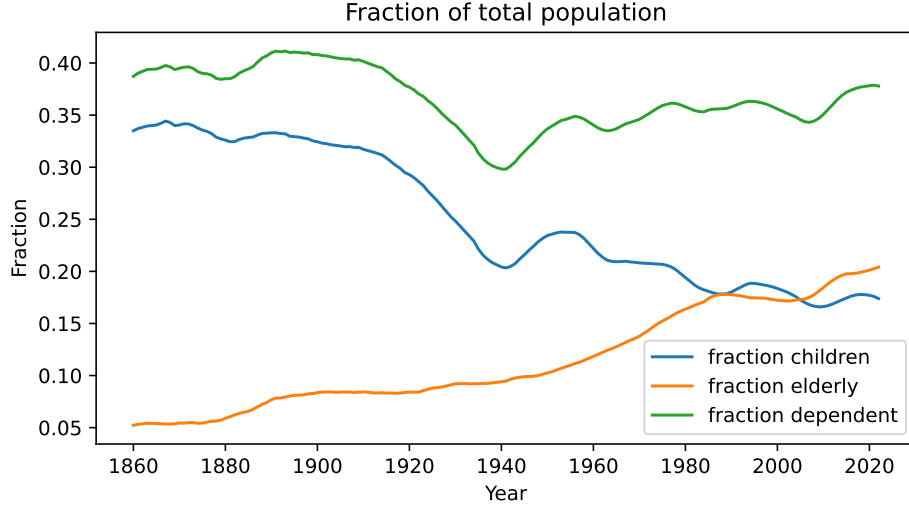


Figure 2: Fractions

Discussion

Advances in healthcare, improved living conditions, and better access to medical services have contributed to increased life expectancy in Sweden. Longer lifespans result in a larger proportion of the population falling into the elderly category, leading to a shift in the age distribution. This is evident in Figure 2, where the fraction of the elderly consistently increases over the entire time span.

Simultaneously, there is a noticeable declining trend in the fraction of children. This phenomenon is attributed to the decreasing fertility rate in Sweden since the 1960s. Various factors contribute to this decline, including postponed marriage, heightened focus on education and career pursuits among women, and evolving societal norms.

Furthermore, a distinct dip in the fraction of children is observed around 1940, likely attributable to the impact of the Second World War.

Many industrialized countries share common demographic challenges, such as aging populations, low fertility rates, and the need for immigration to offset demographic imbalances, Eurostat [2]. These trends have implications for social welfare systems, healthcare, and economic sustainability.

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/.
- [2] Statistics Explained Eurostat. *Population projections in the EU*. Retrieved 2024-01-20. 2023. URL: <https://ec.europa.eu/eurostat/statistics-explained/index.php?oldid=497115>.
- [3] Steven S Skiena. *The Data Science Design Manual*. Retrieved 2024-01-20. 2024. URL: <https://ebookcentral.proquest.com/lib/gu/detail.action?docID=6312797>.

Appendix: Source Code

```
1 import numpy as np
2 import pandas as pd
3 from matplotlib import pyplot
4
5
6 ratio = lambda children, elderly, labor: 100 * (children + elderly)
7         ↪ / labor
8 total = lambda children, elderly, labor: children + elderly + labor
9 fraction = lambda part, total: part / total
10
11 population = pd.read_csv('swedish-population-by-year-and-sex-1860
12         ↪ -2022.csv', sep=',')
13
14 # Drop sex column, we don't need it
15 populationNoSex = population.drop(columns=['sex'])
16
17 # Set age to numeric
18 populationNoSex.at[220, 'age'] = 110
19 populationNoSex.at[221, 'age'] = 110
20
21 # Convert to numeric
22 populationNoSex['age'] = pd.to_numeric(populationNoSex['age'],
23         ↪ errors='coerce', downcast='float')
24
25
26 # Group by age
27 classes = populationNoSex.groupby(pd.cut(populationNoSex['age'],
28         ↪ [-1, 14, 64, 110])).sum()
29
30 # Drop age column, we don't need it anymore
31 classes = classes.drop(columns=['age'])
32
33 # Transpose
34 classesT = classes.transpose()
35
36 # Apply lambda functions
37 classesT['ratio'] = classesT.apply(lambda row: ratio(row.iat[0],
38         ↪ row.iat[2], row.iat[1]), axis=1)
39 classesT['total'] = classesT.apply(lambda row: total(row.iat[0],
40         ↪ row.iat[2], row.iat[1]), axis=1)
41 classesT['fraction_children'] = classesT.apply(lambda row: fraction
42         ↪ (row.iat[0], row.iat[4]), axis=1)
43 classesT['fraction_elderly'] = classesT.apply(lambda row: fraction(
44         ↪ row.iat[2], row.iat[4]), axis=1)
45 classesT['fraction_dependent'] = classesT.apply(lambda row:
46         ↪ fraction(row.iat[0] + row.iat[2], row.iat[4]), axis=1)
47
48
49 # Convert index to float
```

```

35 years = np.asarray(classesT.index.values, float)
36
37 # Plot ratio
38 fig1, ax1 = pyplot.subplots(figsize=(5, 2.7), layout='constrained')
39 ax1.plot(years, classesT['ratio'], label='ratio')
40 ax1.set_xlabel('Year') # Add an x-label to the axes.
41 ax1.set_ylabel('Ratio') # Add a y-label to the axes.
42 ax1.set_title("Dependency-Ratio") # Add a title to the axes.
43 ax1.legend()
44
45 # Plot fractions
46 fig2, ax2 = pyplot.subplots(figsize=(5, 2.7), layout='constrained')
47 ax2.plot(years, classesT['fraction_children'], label='fraction-
    ↳ children')
48 ax2.plot(years, classesT['fraction_elderly'], label='fraction-
    ↳ elderly')
49 ax2.plot(years, classesT['fraction_dependent'], label='fraction-
    ↳ dependent')
50 ax2.set_xlabel('Year') # Add an x-label to the axes.
51 ax2.set_ylabel('Fraction') # Add a y-label to the axes.
52 ax2.set_title("Fraction-of-total-population") # Add a title to the
    ↳ axes.
53 ax2.legend()

```