# DAT565/DIT407 Assignment 1

Ola Bratt
ola.bratt@gmail.com

Patrick Attimont
patrickattimont@gmail.com

2024-01-16

This paper is addressing the assignment 1 study queries within the *Introduction to Data Science & AI*, DIT407 course at the University of Gothenburg. 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 Dependecy Ratio of Sweden from 1860 to 2022 is show. 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).

$$Dependency\, ratio = 100 * \frac{children + elderly}{labor\, force} \tag{1}$$

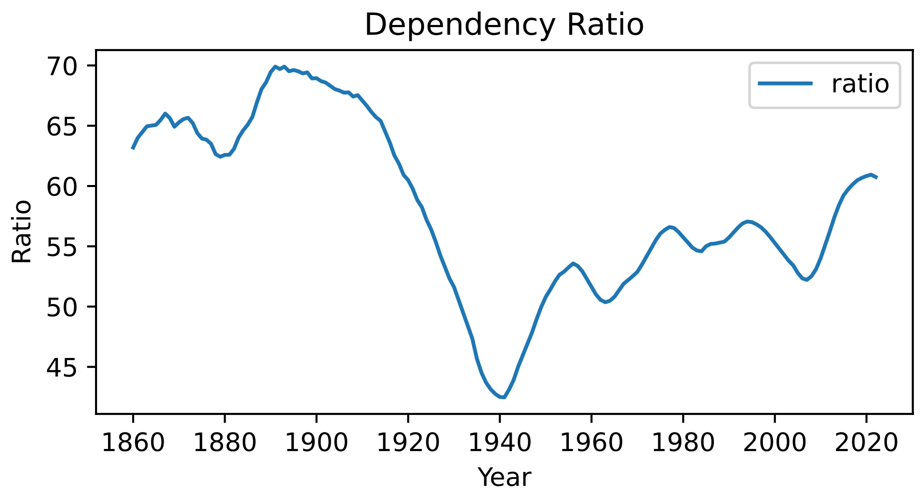

Figure 1: Dependecy ratio

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].

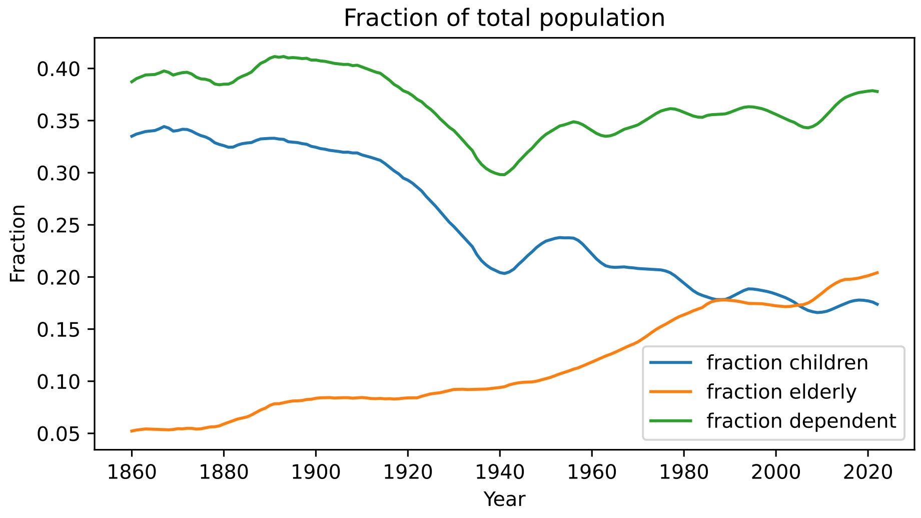

Figure 2: Fractions

## Conclusions

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

```python
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', downcast='float')


# Group by age
classes = populationNoSex.groupby(pd.cut(populationNoSex['age'], [-1, 14, 64, 110])).sum()
# 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], row.iat[1]), axis=1)
classesT['total'] = classesT.apply(lambda row: total(row.iat[0], row.iat[2], row.iat[1]), axis=1)
classesT['fraction_children'] = classesT.apply(lambda row: fraction(row.iat[0], row.iat[4]), axis=1)
classesT['fraction_elderly'] = classesT.apply(lambda row: fraction(row.iat[2], row.iat[4]), axis=1)
classesT['fraction_dependent'] = classesT.apply(lambda row: fraction(row.iat[0] + row.iat[2], row.iat[4]), axis=1)

# Convert index to float
```

```python
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()
```