Python project

Introduction

This project aims to analyze a selection of statistical data, to produce an analysis through visualizations of the data. Based on the visualizations, a conclusion will be reached as to the trends within the data. Understanding trends is useful for predicting future outcomes, and in political contexts for creating policies and setting expectations.

This project consists of this report (.pdf), data files (.csv), a python file (.py) containing the functions which produces the visualizations shown in the **Results** section, and a Jupyter Notebook (.ipynb) which calls the functions on the data. This report covers the content and structure of the selected data, and how the latter impacts the code. In addition, it covers the functions in the written code, the visualizations produced by the code, and conclusions drawn based on the trends within the data.

The focus of this project is on appropriately storing the selected datasets, and loading them into Python in the appropriate structure using the modules Numpy and Pandas, to visualize them using the module Matplotlib.

Data

For this project, the data chosen for analysis is sourced from Danmarks Statistisk, which stores large amounts of statistical information about the country of Denmark and its population. There is an extensive selection of datasets available through their website, too many to include in this project, and thus a small selection has been chosen for this analysis. Below is a list of the selected datasets:

Election to the Parliament by candidates, political party and time (1990-2019)
Hospitalization rates → Population by sex, key figures, group of persons, region, age and time (2006-2018)
Latest use of internet → per cent of the population (16-74 years) by type and latest use (2008-2019)
Members of the National Church → Population 1. January by parish, sex, age and time (2010-2020)

Source: https://www.statistikbanken.dk/10005

Structure

After selecting the data, a file format is chosen for storing it. This is done because storing the data separate from the code, allows the code to function independently of the content of the data, as long as the data files share the same structure.

For the choice of file format, Comma-Separated Values (CSV) was chosen, as it is a commonly used format for storing data. The format itself is simple; each line is a data record consisting of one or more fields, and each field is separated by a comma. Files in this format can be imported into programs such as Microsoft Excel, and displayed as a table of information. Each data record represents a row in a table, and each field represents a column in each row.

Source: https://en.wikipedia.org/wiki/Comma-separated_values

Danmarks Statistisk exports datasets as CSV files, which follow a few additional rules:

- Each dataset begins with a header, which can be multiple lines.
- Following the header, the dataset consists of the columns for its subsets.
- The title of each subset is located between the column and the first subset, and before each subset.
- Each data record starts with a field describing its contents.
- Datasets may contain foodnotes, following its last subset.
- All subsets contain the same amount of data records, fields, and field descriptions.

Functions

This section contains an explanation of the purpose of each function in the python file, including the ways they can be utilized to produce a variety of different results, beyond the results included in this analysis.

Read imports a dataset from a file, into a dictionary of Pandas data frames.

- Loops through the lines in the data file given as input.
  - Strips newlines from each line.
  - Replaces commas within double-quotes with a * symbol, to avoid splitting a field into two fields.
  - Converts the line into a list of fields, ignoring all empty fields.
  - Converts fields with a missing value, which are two periods in the data files, into zeros.
  - Reads the columns of the dataset into a list.
  - Sequentially reads the name and the rows of a subset into a string and a list respectively.
  - Sequentially converts the list of rows into a Numpy array, and converts the array into a Pandas data frame, by using the rows as data and indices, and the list of columns as columns for the data frame.
  - Sequentially adds the name of a subset as a key a dictionary, with the respective data frame as the value.
- Returns the dataset as a dictionary.

Aggregate performs a variety of calculations on a dataset, returning the result as a new dataset.

- Provides the user with a variety of options for aggregating the data in a dataset, and two options for sorting the data from lowest to highest value, or vice versa.
- Checks if the user has given valid input from the available options. If not, returns an error, and does not proceed to aggregate the data.
- If the given input options are valid, it loops through each data frame in a given input dictionary representing a dataset.
  - Transposes the data frame, to process the rows, rather than the columns.
  - Aggregates the data from the columns for each row, depending on the given input option.
  - Sorts the aggregated data frame, depending on the given input option. If the option ‘none’ is given, then the aggregated data frame is not sorted.
  - Adds the name of the data frame as a key to a new dictionary, with the aggregated data frame as the value.
- Returns the new dataset as a dictionary.

Removal removes rows and columns in a dataset, if they only contain zeros.

- Loops through each data frame in a given input dictionary representing a dataset.
  - Indexes and compares the values of a data frame, removing all columns only containing zeros.
  - Indexes and compares the values of a data frame, removing all rows only containing zeros.
  - Adds the name of the data frame as a key to a new dictionary, with the filtered data frame as the value.
- Returns the new dataset as a dictionary.

Combine groups similar data records in a dataset into subsets.

- Creates a list of indices, and a list of columns, from the first data frame in the given input dataset.
- Creates a 3-dimensional Numpy array of the values from the data frames in the dataset.
- Loops through each index from the dataset.
  - Sequentially creates a data frame for each index, with the values from the index in every subset in the dataset, the name of the subsets as the indices, and the same columns as the original data frame.
  - Adds the original index as a key to a new dictionary, with the new data frame as the value.
- Returns the new dataset as a dictionary.

Visualize plots charts for the subsets in a dataset.

- Provides the user with a variety of options for visualizing the data in a given input dataset.
- Checks if the user has given valid input from the available options. If not, returns an error, and does not proceed to visualize the data.
- If the given input option is valid, it loops through each data frame in the dataset.

  - <u>Checks if the input option is for a line chart.</u>
    - Transposes the data frame, to process the rows, rather than the columns.
    - Defines a Matplotlib figure with a length of 15 and a height of 5.
    - Plots the transposed data frame.
    - Defines the legend of the chart as the indices of the data frame, and its location.

  - <u>Checks if the input option is for a scatter chart.</u>
    - Defines a Matplotlib figure with a length of 20 and a height of 5.
    - Loops through each column in the data frame, adding its values to y-axis of the scatter chart, with the x-axis set to the indices of the data frame.
    - Defines the legend of the chart as the columns of the data frame, and its location.

- Checks if the input option is for a stacked bar chart.
    - Defines a Matplotlib figure with a length of 10 and a height of 5.
    - Transposes the data frame, to process the rows, rather than the columns.
    - Creates a list of zeros with the same length as the amount of columns in the data frame. This list is used to stack the data-frame at the bottom of the chart.
    - Loops through each index from the dataset, adding the values for the index to the y-axis of the stacked chart, with the x-axis set to the columns of the data frame. The y-axis values are stacked by adding the values to the list of zeros, at the end of each iteration.
    - Defines the legend of the chart as the indices of the data frame, and its location.

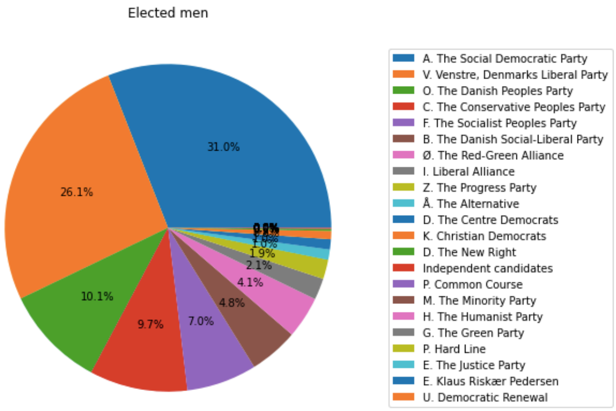
- Checks if the input option is for a pie chart.
    - Defines a Matplotlib figure with a length of 7 and a height of 7.
    - Plots the values of the data frame as a pie chart, with its values on each slice.
    - Defines the legend of the chart as the indices of the data frame, and its location.

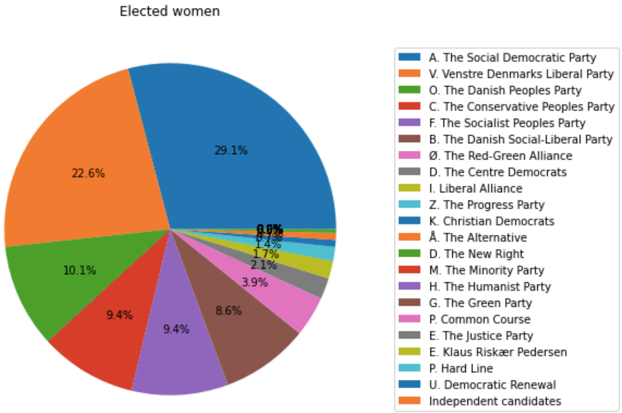
- Defines the title of the chart as the name of the data frame.
- Rotates the labels on the x-axis by 90-degrees.
- Displays the chart in the program.

Results

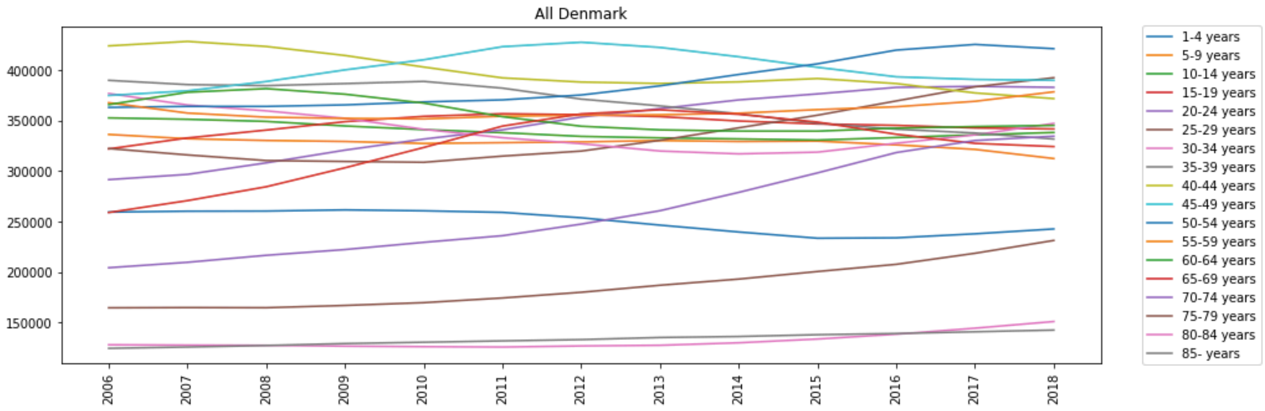
This section gives an overview of the visualizations produced by the notebook. One type of chart has been chosen for each dataset, and different combinations of functions were utilized to produce the charts.

Election results were summed and sorted from highest to lowest value, using the aggregate-function. A pie chart was chosen for this dataset, because it neatly conveys the distribution of seats earned by the parties.



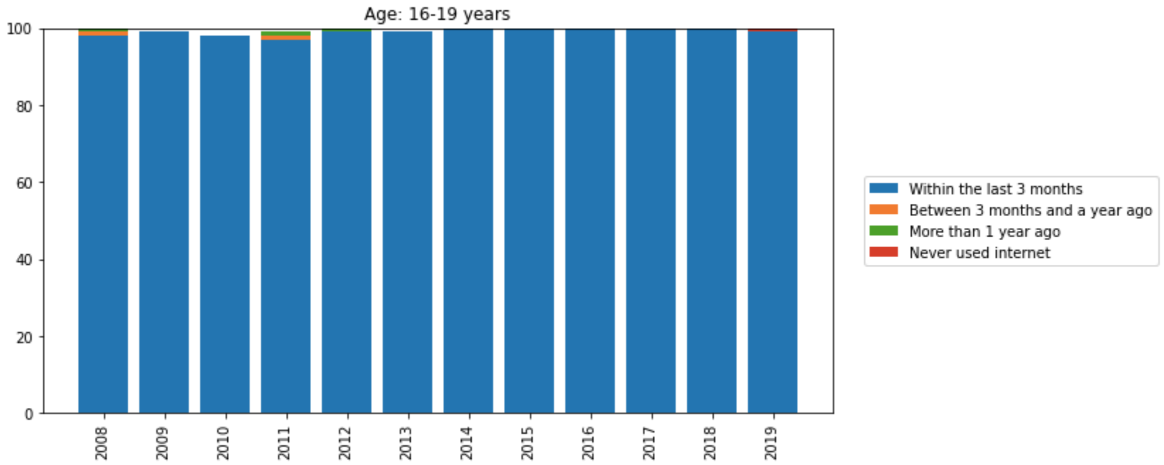


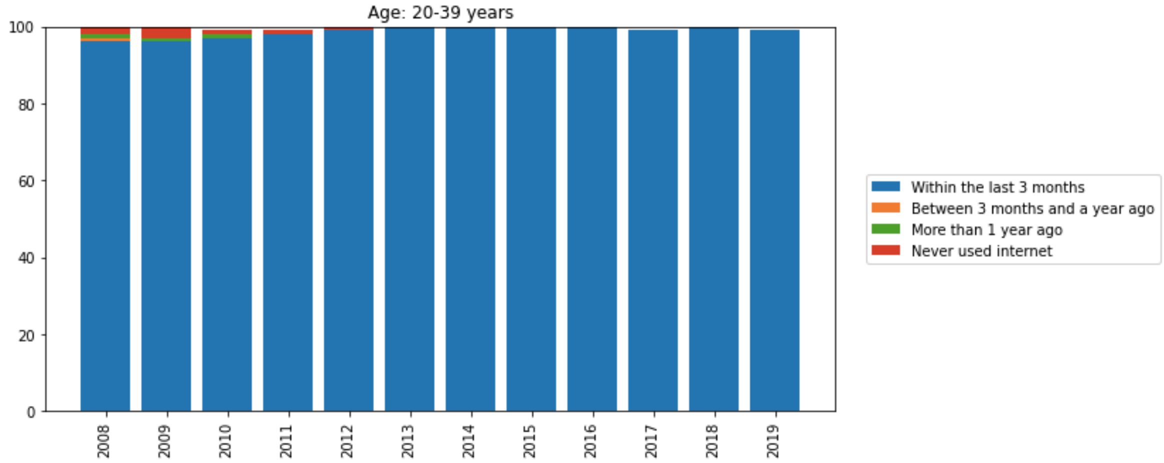
Hospitalization rates were visualized as a line chart, to convey the evolution for each age range over time.

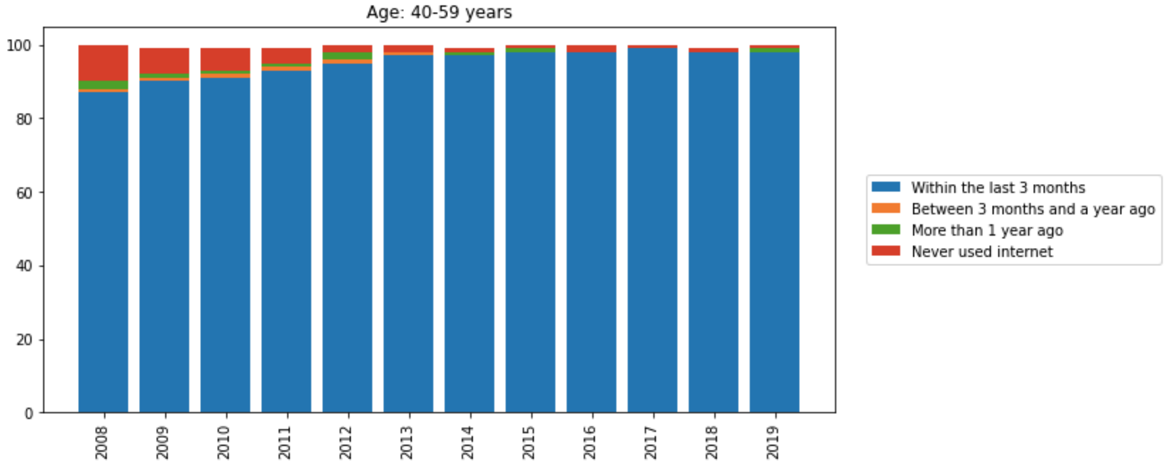


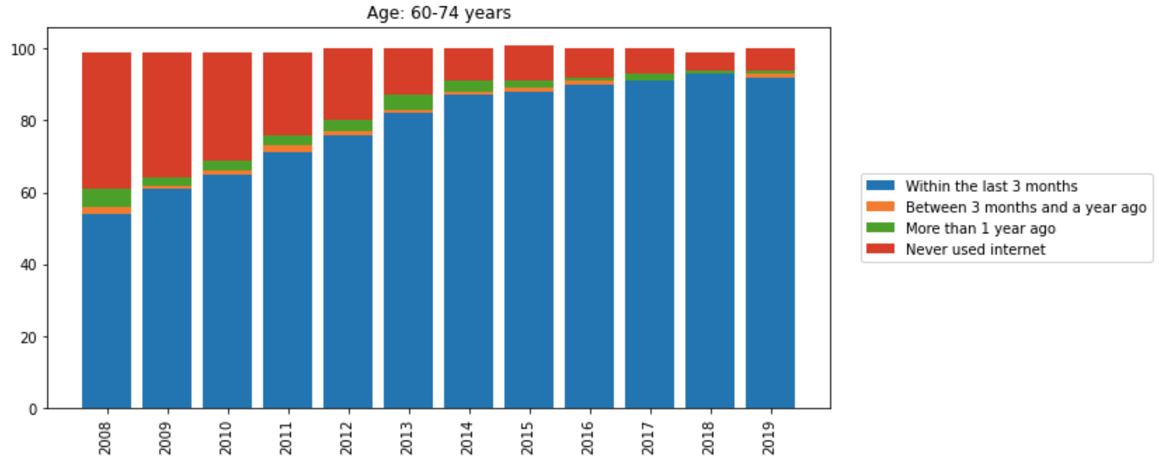
Internet usage data was combined using the combine-function, into a new dataset of subsets for each age group, rather than subsets for each range of usage. Stacked bar charts were chosen for this dataset, to convey the distribution of each range of usage, for each year in the data. Pie charts were not chosen for this dataset, as the amount of columns would result in a large amount of charts, that would not convey the evolution over time.

Note: the usage data is in integers representing percentages, and thus not all of the bars sum to 100%, because some values in the dataset have been rounded up or rounded down.

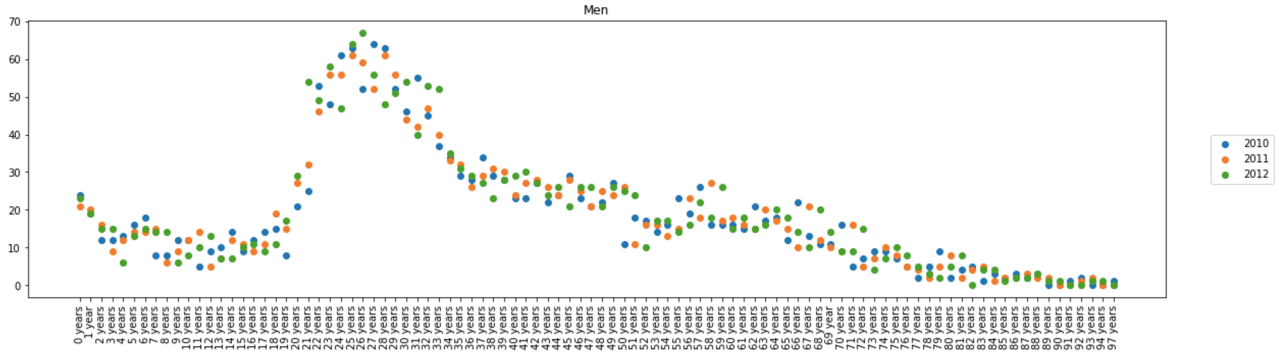

Age: 16-19 years
0
20
40
60
80
100
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
Within the last 3 months
Between 3 months and a year ago
More than 1 year ago
Never used internet

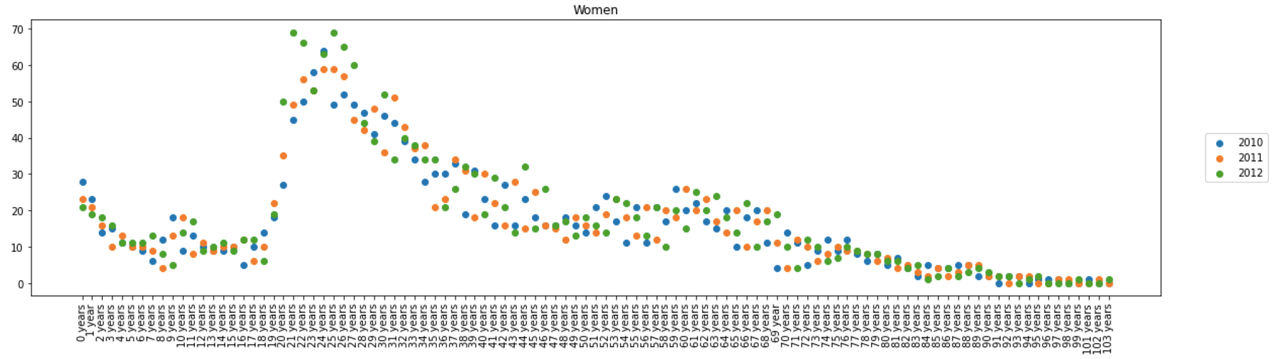

Age: 20-39 years
0
20
40
60
80
100
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
Within the last 3 months
Between 3 months and a year ago
More than 1 year ago
Never used internet


Age: 40-59 years
0
20
40
60
80
100
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
Within the last 3 months
Between 3 months and a year ago
More than 1 year ago
Never used internet


Age: 60-74 years
0
20
40
60
80
100
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
Within the last 3 months
Between 3 months and a year ago
More than 1 year ago
Never used internet

Membership data was filtered using the removal-function, because many of the rows and columns contained no data. A scatter chart was chosen to visualize the dataset, as an example of how a scatter chart would look. None of the chosen datasets fit with this type of chart, so it is purely for demonstrative purposes, and does not convey anything about the dataset.





Conclusion

This section contains the insights obtained through the analysis and the results produced by the code.

Election results visualized as pie charts, show that despite the large amount of parties (22), 57.1% of men and 51.7% of women were from the two parties “The Social Democratic Party” and “Venstre, Denmarks Liberal Party.”

Hospitalization rates visualized as a line chart, show that the elderly population has seen an increase in hospitalization, but that a majority of hospitalizations is of younger people.

Internet usage visualized as stacked bar charts, show a trend towards more usage within every age range, but also that internet usage decreases with age. Almost every person in the age range 16-39 had used the internet within the last 3 months, regardless of the year, while almost 40% of people in the age range 60-74 had never used the internet in 2008, down to less than 10% in 2019.