Universiteit Leiden
The Netherlands

Computer Science & Economics

# Visualisation tools to support historical research on a linked dataset about Leiden University

Liam van Dreumel

Supervisors:
Wessel Kraaij
Richard van Dijk

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

17/08/2022

**Abstract**

The Linking University, City and Diversity project is an ambitious collaboration between different disciplines at the Leiden University. Together, data scientists and historians will work on creating a concrete view of the relationship between the city and its university. The perspectives of interest are the mobility, the geographical segregation, and the integration of former students and professors at the Leiden University. For the historians, the aim of this project is to provide a better qualitative and quantitative understanding of the interaction between city and university. An extension of this aim is to lay the groundwork for further research, one of the possible topics being diversity. The data scientists will use this project as a way to develop better tools and knowledge about structuring, analysing and visualising raw, unstructured data. The research in this thesis evaluates applicable models for data visualisation of a large historic data set. Multiple visualisation methods, such as graphs, geographical maps and network diagrams, were analysed to find appropriate ways to visualise specific historic data. Two prototypes featuring visualisation methods were made that showcase a visualisation approach for the data set. The prototypes showed possible solutions for the visualisation problem. With further development, the prototypes could be improved to offer an insight in historical data for a general audience and allow historians to perform research on historical data sets.

**Acknowledgements**

I would like to thank the following people for helping me realise this thesis and supporting me during my time working on the project:
Wessel Kraaij, for introducing the project, for accepting me as part of team and for guidance during the writing of my bachelor thesis.
Richard van Dijk, for guiding me as project leader and providing me with knowledge and resources to create this thesis and contribute to the project.
Fellow students Rick Schreuder and Michael de Koning, for collaborating with me on parts of the project.
Ariadne Schmidt, Joost Visser, Pieter Slaman and Alicia Schrikker for providing me with feedback about my work and giving me new ideas to research.

# Contents

# 1 Introduction

Digitisation is a growing need in our modern world. Even the history section of the scientific world cannot escape this fact, how ironic it may seem. The inevitability of digitisation also gives us opportunities however. If historical data is digitised, stored in a database and ready to be shown, what are the limits? How can we visualise this data in a way that conveys the message it holds? How can digital humanities profit from such visualisation? This thesis will evaluate these questions and give concrete options of visualisation for a large historical data set.

## 1.1 Linking University, City and Diversity project

The linking UCD project is an ambitious collaboration between different disciplines within the Leiden University[Ariadne Schmidt and Wessel Kraaij, 2021]. Together, data scientists and historians will work on creating a concrete view of the relationship between the city and its university. A study of the relationship between the city and the university should preferably be based on as much historical data as possible. Besides this quantitative desire, the historical data that is shown should provide qualitative insight in the relationship. The relationship between city and university is a broad term. Many perspectives can be studied. The perspectives of interest that the projects focuses on are:

- Mobility
  The mobility of people. Where did they come from? Where did they study? Did they stay after when they were done studying, or did they move to other places? How did this impact the city?

- Geographical segregation
  The segregation of people. Where in the city did they live? Did students live together? Did the professors live segregated? Was there a divide between the academics? Did this divide change over time?

- Social integration
  The integration of people in the city through their enrollment at the university. With whom did they interact? Did they marry in the city? Did they go to work in the city after studying? What effect did this integration have on the city?

Part of the project is diversity. Walking through the academic premises might give the illusion that the university was mostly represented by one group of people. However, the academic community was already diverse in the last few centuries[Schmidt, 2021]. An important part of the project is therefore to show the differences in backgrounds of the students and professors attending the university. The term diversity can encompass many aspects of a person's background. The aspects of diversity that are covered in the visualisation of this thesis are limited to the geographical-, gender-, religious-, and social-economic backgrounds of historical persons.
To bring these questions to life, the historians collaborate with data scientists to digitise historical sources and create links between different data types and sources. The data types are incorporated in a data model. The data in the model is visualised in a visualisation framework. The data that is used is provided by historical and heritage institutions. Most interesting are the linking of documents regarding population information with actual people and locations. Think of enrollment

and registration lists, population registers, census, birth and marriage certificates and land registry data. With people and places linked, historians can research the movement of groups of people through time, where and when they lived and where they moved to.

### 1.1.1 Goals

The end goal of the project is to show the achieved results at the Leiden City of Science 2022. The Leiden City of Science 2022 is a year long event where the city will host a multitude of activities regarding various disciplines in the scientific world[Universiteit Leiden, 2021]. The following goals should be reached to come to the end result:

A storage component must be made. In this storage component, users can add useful historical entities that they have found to the project. Another import feature would be the possibility for users to link the entities with each other. Bachelor student Rick Schreuder created a data model in the form of a database as a member of the LUCD project.

In combination with the database, adapters have to be created. These adapters are used to modify data that is difficult to add manually to the database. The adapters can verify, clean, link and enrich new data, change the structure if necessary and then insert the result into the database. A second part of the adapters is the frequent automatic importing of new data from various external sources into the database. Bachelor student Michael de Koning worked on these adapters as part of the LUCD project.

The data in the database has to be visualised in order to be viewed by both historians and a general audience. The visualisation will be in the form of a website that can be used by historians and other interested parties to research and learn more about the history of the city. Data inside the database will be send to the visualisation components to be formed into graphs, texts and interactive means.

A workflow component is needed that controls the flow of data. The workflow receives requests to send and or receive data from the other components. The workflow would make sure that the requested data goes to the correct components.

### 1.1.2 Project plan and thesis objectives

The LUCD project will lay the groundwork for future historic research about the relation between city and university. Some work has been done on digitising historic sources, consequently work will be done on text mining analysis on historic newspapers from the Koninklijke Bibliotheek Den Haag [Ariadne Schmidt and Wessel Kraaij, 2021]. Three bachelor students joined the project in order to work on the first iterations of the database, the data adapters and the data visualisation. After their contribution has finished the project team will continue based on their works. The three bachelor students include the aforementioned Michael de Koning, Rick Schreuder and myself. This thesis is concerned with the visualisation part of the LUCD project. The objective of this bachelor project was to create possible visualisation prototypes that can be integrated in the future LUCD web page. The prototypes consist of multiple visualisation styles that use the available historic data. The following steps have been taken to come to the final product:

- Collecting requirements for the visualisation
- Research of historical data presentation
- Analysis of visualisation methods and software
- Building prototypes with implementation of data visualisation methods

The goal is to make the completed prototypes accessible to the public through a website. The data that is used for the prototypes is based of two excel sheets gathered by the project team, and an excel sheet containing data scrapped from Wikipedia by myself. After the creation of the website, the data sources are to be replaced by connecting the website with the database. The database contains the data present in the excel sheets.

## 1.2 Research question

The following research question will be answered in this thesis:

*What are applicable software frameworks for the creation of interactive visualisations of a large historical data set about the scholars and students of Leiden University in the past 450 years.*

The research question states the visualisation problem of this thesis. Some sub questions have been formalised to create answers that will help solve the main visualisation problem and creates a structure for this thesis to evaluate these questions.

- Sub question 1: Which data present in the available data sets should be visualised?
  The provided data offers information about people. Therefore, the historical entities this project will consist of are persons. The next sub question that arises is:
  - Sub question 1.1: Which characteristics of the historical entities are relevant to visualise?
    The provided data sets offers multiple characteristics for every person. The characteristics that will be visualised should fit the research questions of the mobility, geographical segregation and social integration perspectives.

  Section 3 evaluates the available data and what parts of it can be used for the visualisation. This will provide answers for sub question 1. The research questions are listed in section 4.

- Sub question 2: How to visualise the data?
  The chosen set of characteristics of a historical entity need a form of visualisation to requirements in the form of research questions. These questions were obtained from requirement interviews. Sections 2 and 5 will evaluate existing visualisation methods and works that have dealt with similar visualisation problems. Knowledge of previous solutions and available methods will help create ideas to solve this sub question. The collected research questions are detailed in 4.

- Sub question 3: How to implement interactivity in the visualisation?
  Interactivity is one of the key elements of the visualisation for two reasons, depending on the user. Historians and researchers who will want to use the final product need means to interact

with the data. They need to change inputs and views in order to research the data and get a clear view of their current subject. The other user base consists of enthusiast and other interested parties. More engagement between the audience and the data should create more interest in the final product. Previous work on interactivity in data visualisation is discussed in section 2.2. Section 4.2.2 details software requirements regarding visualisation. Furthermore, sections 5 and 6 look at interactivity options in visualisation methods and programs.

- Sub question 4: What type of visualisation methods are applicable to the data?
  A set of attributes can be created after knowing what kind of data to visualise and how to present it. A visualisation method should adhere to these attributes in order to be a useful means of visualisation. After compiling a list of possible visualisation methods they will be tested against the set of attributes. The degree of interactivity also depends on the types of visualisation. Therefore, interaction methods is another requirement when selecting applicable visualisation methods. The methods that fit the requirements the most will be used in the development of the final product.
  The visualisation methods section 5 goes through possible methods that can be applied to the data.

- Sub question 5: What applicable software supports the chosen visualisation methods?
  There are many different ways of presenting visualisations. A variety of commercial software and frameworks are available. The possible visualisation methods and means of interactivity depend on which software is chosen. One of the possibilities for the LUCD project team is using a Django framework to host their website. Connectivity with this framework is another requirement for the program. Only free solutions are being considered in this thesis since there is no monetary budget available. Paid features in software will be considered, but not implemented in the final product.
  Possible options for sub question 5 are mentioned in section 6.

The sub questions are handled in sections 2 through 7.1. The answers of the sub questions are used to structure and build the final prototypes in sections 7.2 and 7.3. The final prototypes are used to answer the main research question.

## 1.3 Thesis overview

This thesis is structured as followed: The section related work 2 covers papers and research covering comparable data visualisation questions and similar projects that visualise historic data. The provided data that has been used is detailed in the data section 3. The requirements gathered from the requirement interviews are listed in section 4 The section visualisation methods 5 explains all methods of visualisation that have been researched. Section 6 covers the assessment of software. The implementation of two visualisation prototypes is detailed in section 7. Some final notes, limitations and further research will be discussed in the discussion section 8. Lastly, the conclusion is written in section 9.

# 2 Related Work

This section will cover previous work done on the topic of data visualisation. Data visualisation is used to make a visual representation of data. Converting raw data into understandable graphs and charts is useful because it would be impractical for people to go through all the raw data and draw conclusions themselves. Especially when people lack the proper skills to interpret data in a meaningful way. Because of its graphical nature, visualised data can be made appealing to entice users. Visual representation can grab the attention of potential viewers and might be a more effective way of presenting data compared to a textual presentation. Research at the university of Pennsylvania claims that the human eye can transmit visual data at roughly the speed of an Ethernet connection [Koch et al., 2006]. Furthermore, according to research done by a visual marketing platform, 90 percent of information transmitted to the brain is visual and visual information is processed 60.000 times faster than textual information is [Pastore, 2020].

## 2.1 Rules for data visualisation

Sub question 2 is concerned with how to visualise data. Besides knowing which parts of the data need to visualised, it is also important to know what the best practices are in regards to showing this data. Tufte [Tufte, 1983], wrote a number of principles regarding data visualisation. The first six rules describe the graphical integrity of a graph:

1. *The representation of numbers, as physically measured on the surface of the graphic itself, should be directly proportional to the numerical quantities measured.*
   The numbers shown in graphs need to be properly scaled in order to not misrepresent the measured difference between the numbers.

2. *Clear, detailed, and thorough labeling should be used to defeat graphical distortion and ambiguity. Write out explanations of the data on the graphic itself. Label important events in the data.*
   Labels should give clarity to the data that is shown in the graph. Data without context cannot be interpreted.

3. *Show data variation, not design variation.* It is better to have a consistent and clear design instead of creating numerous complicated graphs that take a while to figure out.

4. *In time-series displays of money, deflated and standardized units of monetary measurement are nearly always better than nominal units.*
   Numerical data should be displayed in units to give them meaning. Furthermore, the units should be standardised in order for all viewers to understand them.

5. *The number of information-carrying (variable) dimensions depicted should not exceed the number of dimensions in the data.*
   A graph should not have more dimensions than the data has, otherwise the added dimension would not have a purpose since they are not present in the data.

6. *Graphics must not quote data out of context.*
   A graph should reflect the data and not be twisted to give a misleading perspective.

In his work, Tufte also described a few definitions regarding data visualisation. The following sections will cover these definitions.

### 2.1.1 Data-Ink

In chapter 4, Tufte spoke about data-ink [Tufte, 1983]. And even though most data visualisation is online and doesn't require ink, the principles still stand. The majority of the graph should consist of data-information. It is the core of the graphic that cannot be removed without the graphic losing it's meaning. This is called the Data-Ink. The important thing to learn from these principles is to prioritise the showing of data and minimise the amount of non-data graphics within the figure. Keep working on the graph until it fits these criteria.

### 2.1.2 Chartjunk

Chartjunk, as Tufte calls it in chapter 5 of his work, is the use of excessive and unnecessary graphical effects within a graph [Tufte, 1983]. Graphs should showcase the data, not the design of the graph. Attractive and attention-pulling designs can draw viewers into the graph, but without substantial data, the viewers won't learn anything and the graph would be without purpose.

### 2.1.3 Data Density

The data density is the ratio of the part of the graph that displays the data and the total size of the graph. Tufte uses the following formula: data density of a graphic = number of entries in data matrix / area of data graphic [Tufte, 1983]. The data matrix is the area of the graph that includes the observations from the data. A higher data density graph consists of mostly data and less of anything else. In order to maximise the data density, the graph can be shrunk down as much as possible without losing any of the crucial information. According to the Shrink Prinicple, Graphics can be shrunk way down. According to the prinicple, many graphs can be reduced to half their size without losing any legibility and information.

### 2.1.4 Small Multiples

Small multiples are a series of frames of a graph. Every frame is a separate graph consisting of the same data, where one of the variables changes per frame. E.g. a series of graphs could show the wheather conditions of the Netherlands over 24 hours, with 24 graphs consisting of a map of the Netherlands spanning one hour. The design remains consistent for every graph. This results in a focus on the data and its changes per graph.

## 2.2 User interactivity with digital data visualisations

Interactivity of visualisation is the main concern of sub question 3. There are multiple platforms for a data scientist to create interactive code and figures like Plotly and the Code Ocean platform. Some journals and publishers have options for Code Ocean integration which helps users see interactive images instead of static pictures [Perkel, 2018]. The Code Ocean platform is a no-code computational workbench where users create, develop and share data analysis [Ocean, 2022]. Plotly supports many types of graphs that users build and share. All Plotly graphs feature interactive

options like zooming in- and out. Plotly is usable in multiple programming languages like Julia, MATLAB, Python and R. Binder is a tool that creates custom computing workspaces wherein Jupyter Notebook and R code can be run. These workspaces can be shared with other researchers. A tool like Binder helps with reproducibility and simplifies peer reviewing [Perkel, 2018]. The previously mentioned Jupyter Notebook is a web application where users create computational documents that can be shared [Jupyter, 2022a]. The company introduced JupyterLab, the new version of notebook. It features over 40 programming languages, interactive output and integration of big data tools. JupyterLab can accessed through other code editing software as Visual Studio and Pycharm as a plugin. Plotly's data app Plotly Dash gives users the option to build data dashboards [Enterprise, 2022]. These dashboards can function as analytical tools can be used as models to research data from specific problems. High quality 3D graphics of the globe are being used for data visualisation of the Earth. These graphics allow users to view the planet in detail. Applications of geographical visualisation will be further discussed in section 5.1. The JavaScript API WebGL allows 2D and 3D graphics to be rendered in browsers. This gives users the ability to feature interactive 2D and 3D graphics on their web pages. These images are not limited to geographical content and can be expanded into every field. Analytical tools like the web-based SPOT lets users perform interactive analysis on their data [Diblen et al., 2019]. The tool supports integration with a PostgreSQL database.

## 2.3 Visualisation of historical data

There are many forms of data visualisation. Some of these methods that will be applicable to the LUCD visualisation will be covered in the visualisation methods section 5. This section will evaluate some examples of visualisation of historical data to find the visualisation methods that are commonly used.

### 2.3.1 Canon of Leiden

The website Canon of Leiden features information about monuments, people and events in the city of Leiden [Blom, 2008]. The canon features: A timeline, divided into seven coloured time periods that each have their own page of information; A separate page with individual objects. Every object is part of one of the seven time periods. The objects also have their own pages; A map of the city. With every object is placed on the map, corresponding to their relevant location.

### 2.3.2 Historic Leiden on the map

The project Historic Leiden on the map aims to gather as much historic geographical information about the city of Leiden [leiden Omstreken, 2008]. Data from different sources is merged together to create detailed maps of the city. The multiple maps on the website show highlighted plots of lands that can be clicked on to see more information about the plot and its owners in that time. Figure 2 shows an example of one of the historic maps. The website features a database that can be searched for specific plots, people or certificates.


(a) Timeline of the city of Leiden Source: Canon of Leiden,
$http://www.deleidsecanon.nl/index.php?option=com_content&view=article&id=2:testcat&id=3:info$


(b) Map with historic objects of Leiden. Source: Canon of Leiden,
$http://www.deleidsecanon.nl/index.php?option=com_content&view=article&id=3&Itemid=93$

Figure 1: Canon of Leiden website

Figure 2: Map of Leiden in 1543. Source: Historisch Leiden in Kaart,
$https://historischleideninkaart.nl$

### 2.3.3 Dutch Canon

The dutch canon offers an interactive overview of the handpicked important events throughout the history of the Netherlands [van Nederland, 2022]. Every item in the canon features a combination of textual, image, audio and video information. Users can quickly browse through the items in chronological order. A timeline is present at every point to remind users of the time they are currently exploring. Figure 35 shows the lay-out of the Dutch canon.

### 2.3.4 KB Lab

KB Lab is a research department of the Koninklijke Bibliotheek of the Netherlands [Lab, 2022]. The website offers experimental visualisation tools that work with historical data sets. E.g: Timelapses of Dutch newspapers [Faber, 2021] 36; Narralyzer, a tool that creates relationships between characters

in texts [Wildschut, 2017] 37; SIAMESE, a neural network tool that searches in newspapers for images with similar visual trends from a source image [Lonij, 2017] 38; Fame generator, creates a network diagram that collects the most important words from a text to give a meaningful representation of the original source [Lonij, 2017] 38; A tool that counts the amount of images in newspapers, the percentage of pages with images and where in the data sets to find the images [Faber, 2015] 40

### 2.3.5 Delpher

Delpher is a website that allows users to find texts from digitalised dutch newspaper, books and magazines from multiple databases [Delpher, 2022]. Every word in the files are readable and thus searchable. Figure 41 contains a result in a newspaper from the word 'brand'.

### 2.3.6 WieWasWie

WieWasWie is a genealogical database that features person information based on document sources [CBG, 2022]. The website's main purpose is family history research. Searching a name results in names that are linked to documents. E.g. searching the name 'Jansen' brings up a death certificate of a person with that name. Family relations based on the documents are shown. A link to the document is provided. Figure 42 show an example of a name search and death certificate result.

### 2.3.7 FactGrid

FactGrid is a Wikipedia installation open database that can be used to find research data [FactGrid, 2022]. Users can look for and contribute data to the database and can be accessed in many languages. Researchers can add their own data to the database that other research projects can work with. Data can be downloaded to be used in other projects. Users have the freedom to create their own database objects and structure them in ways that work for their specific project. Data can be accessed through the use of SPARQL queries. Data is by writing and searched by combining objects (Q numbers) and relations (p numbers). E.g. Q312 could be a family name called 'Johnson', this is an object. P247 could a relation that states 'family name'. Combing P247 and Q312 means: a family name, called 'Johnson'. Quering these numbers results in objects with the family name 'Johnson'.

### 2.3.8 Leiden University

The Leiden University has its own collection of former professors [University, 2022]. The web page features a list of former professors who each have their own separate page that is available for viewing after selecting them. Users can use search terms to filter the list or select the categories that they want to apply to their search. Figure 43 shows the web page. One of the data sources used in this thesis contains a subset of the professors data present in the web page.

# 3 Data

Three data sets have been used for the visualisation. Both sets were compiled as .xlsx files. All the data in the files is structured.

## 3.1 Professor data

The professor data consists of 1181 entries of former professors at the Leiden University. The data ranges from the 16th century to the present day. Besides some standard information, the data set contains 4 duplicate sets of columns with information detailing the job term at the university. The reason for the duplicate columns is presumably to allow for the recording of multiple job tenures. It is possible for professors to have worked at the university for multiple periods. During these periods, the professors could have worked at different departments teaching different subjects. All duplicate columns with the same subject share the same column name, distinguished by a roman numeral (I - IV) to denote to which set they belong.

Barring the duplicate columns, the following unique columns are present in the data set:

- General information:
  ID, Nobel price, Last name, First name, Preferred name, Gender, Title, Date of birth, Place of birth, Country of birth, Date of death, Place of death, Country of death, Promotion type, Institution of promotion, Date of promotion, Dissertation of promotion
- Job term information (I - IV):
  Profession title, Subject area(s)*, Date of appointment, Date of job acceptance, Teaching subject(s)*, Date of oration, Oration subject, Faculty, End of employment date, Reason end of employment, Special remarks, CV (two columns), VIAF, Handle, NTA/PPN

*All columns accompanied with the asterisk can have multiple entries.

Not all characteristic will be relevant for the project. And those that are relevant might not be for all of the questions (sub question 2.1 - 2.3) stated in section 1.2. The tables 6 and 7 illustrate this relevancy of the columns for the thesis questions. An extra fourth column is added to the table to show which characteristics are relevant for the description of persons. Only characteristics with at least one area of relevance are shown in the figure.

## 3.2 Student data

The student data set covers student enrollments at the Leiden University from the year 1575 to 1812. The student data set has significantly more entries compared to the professor data set. Every row contains general information regarding the student combined with information about the enrollment into the university. The data set has been created by historian Martine Zoeteman and used in her dissertation about the student populace in Leiden from 1575 to 1812 [Zoeteman, 2011]. Both the original and translated names of persons and places are available in the data set. This thesis only handles the translated names that coincide with the modern versions of the names we use today.

With the exception of the columns with original names, the following columns are present in the data set:

- General information
  First name, Last name, Place of origin, Region of origin, Country of origin, Continent of origin, Birth year, Complement, Remark, Title, Royal status, Job, Religion
- Enrollment information
  Enrollment date, Enrollment age, Faculty, Times enrolled, Year of previous enrollment, Faculty of previous enrollment

Tables 8 and 9 state whether the characteristics are relevant to the thesis questions or as general personal information. Characteristics that bore no relevance at all have been omitted from the tables.

## 3.3 Rectores Magnifici data

Through the use of a scraper, a list of all the Rectores Magnifici of the Leiden University has been compiled into a .csv file. The following information about the rectores has been gathered: Term period, Full name, Picture url, Local picture storage location, Details/Term.

## 3.4 Missing and inconsistent data

The data sets are not complete. Table 10 gives a summary of missing entries in the professor and student data sets. The professor data set had inconsistencies in all the columns with dates. Multiple different date formats were present in these rows. All dates have been updated to a consistent D/M/Y format. Not all data entries have complete dates. Some entries were provided with estimates or only contain part of the data e.g. a year. The most notable omissions in the students data is the lack of birth and death dates and places, and gender. The lack of this data limits the possibilities of the geographical chronology and mobility questions. The data set has other missing data in the form of empty entries.

# 4 Requirements

During the course of the project, multiple talks have taken place to discuss the progress of the thesis work. Stakeholders of the project were present in some of the meetings. The stakeholders included historians and data scientists working at the Leiden University. Discussions in these meetings were about the available data, possible new data sources and how to visualise the data. Earlier builds of the prototype were shown to the stakeholders. This led to more feedback regarding what data to visualise and how to visualise it. The requirements that were collected from the meetings are detailed in the functional requirements section 4.1. The non-functional requirements are listed in section 4.2

## 4.1 Functional requirements

The requirements gathered through the requirement interviews have been transformed into research questions with corresponding data sources and visualisation methods. Each research question falls under one of the perspectives of interest that were explained in section 1, that being mobility, geographical segregation and social integration.

### 4.1.1 Mobility

| Research question | Data source | Visualisation method |
|---|---|---|
| Where did they come from? | professor excel file (1575-now), student excel file (1575-1812), english students file | timeline, geographical map |
| How long did they stay in Leiden? | certificates, Erfgoed Leiden | graph, timeline, geographical map |
| Where did they go to after studying? | certificates, Erfgoed Leiden | timeline, geographical map |
| What is there main occupation? | HISCO, IISG | graph |
| What did they study/teach? | professor excel file (1575-now), student excel file (1575-1812), english students file | graph |

Table 1: Mobility of students and professors

Mobility focuses on both chronological and geographical information. The chronological aspect can be visualised through timelines that show the coming an going of people over periods of time. The geographical aspect can be visualised through maps showing where people came from and where they went.

### 4.1.2 Geographical segregation

Geographical segregation is solely focused on creating a geographical representation of the living situation of people in Leiden. The corresponding visualisation would consist of city maps of parts

| Research question | Data source | Visualisation method |
|---|---|---|
| Where in Leiden did they live? | census Erfgoed Leiden | geographical map |
| Did students live together? | census, Erfgoed Leiden | geographical map |
| Did professors live segregated? | census, Erfgoed Leiden | geographical map |
| Was there a divide between academics? | census, Erfgoed Leiden, professor excel file (1575-now), student excel file (1575-1812) | geographical map |

Table 2: Geographical segregation of students and professors

of Leiden detailing in which houses people working or studying at the university lived during that time.

#### 4.1.3 Social integration

| Research question | Data source | Visualisation method |
|---|---|---|
| Who did people befriend and marry? | gentleman's books and directories (1816-1889), certificates, appeal decisions Vierschaar (1658-1810) | (network) graph, family tree |
| Who were there parents and offspring? | Erfgoed Leiden | family tree |
| Did they marry in Leiden? | certificates, Erfgoed Leiden | (network) graph, family tree |
| Did they work in the Leiden? | certificates, Erfgoed Leiden | (network) graph, geographical map |

Table 3: Social integration of students and professors

The social integration contains mostly personal relationships. These relationships can be visualised through network graphs, family trees being a specific form of a network graph. People in the same family, marriage or work fields can be grouped together in networks.

#### 4.1.4 Status Rector Magnificus

| Research question | Data source | Visualisation method |
|---|---|---|
| What is their status? | Wikipedia | table |
| What specific characteristics do they poses? | Wikipedia | table, picture |
| What are their family ties? | Erfgoed Leiden | family tree |

Table 4: Status of rectores magnifici

The rector magnificus is a special position at the Leiden University. This position was rotated yearly for most of its history and could give a good insight in the university and its history. The

idea to include the rectores came up during a requirement interview with Ariadne Schmidt and Pieter Slaman. This led to the creation of a web scraper that collected the information of the rectores from the rectores Wikipedia page [rec, 2022], including pictures of every rector.

## 4.2 Non-functional requirements

This section goes through the non-functional requirements that were gathered over the course of the project.

### 4.2.1 Programming language

Python was the primary programming language for the creation of the visualisation frameworks. Python was chosen for its resources regarding data visualisation. More information about these resources is covered in section 7.1. Other reasons for choosing Python were the familiarity with the language, and the knowledge about the visualisation capabilities of the language through previous projects with the language. In these previous projects, Python was used for data visualisation.

### 4.2.2 Software requirements

The requirements used for the selection of the visualisation software are listed in section 6. General software requirements for the developed prototype were:

- The lay out choice of the prototype. The lay out of the prototype should be neutral and not distract from the visualisation elements.
- The speed of the prototype. A focus was to reduce loading speeds of the visualisations where possible in order to increase user experience. Therefore, loading speeds of the prototype pages should not take longer than a few seconds

The following requirements concerning interactivity where formed:

- Functional interactivity. This means that the user should be able to have a high level of customisability to create a personalised view. This includes interactivity tools that can be used to perform historical research on the data.
- Another requirement is user friendliness. All customisation options should be clear to see and intuitive to use. This would increase the user experience.
- The Aesthetic interactivity. Aesthetic interactivity is meant to increase user enjoyment while using the final prototype. This type of interactivity is focused on creating unique and eye-catching interactions that attract users and keep them engaged with the visualisations.

# 5 Visualisation Methods

This section describes the possible methods of visualisation. The methods have been chosen based on the functional requirements covered in section 4.1.

## 5.1 Geographical map

A focus point of the project is the geographical presentation. The geographical chronology and mobility questions need a geographical map solution in order to visualise the movements of persons. A static image of a map does not suffice for these questions. The map should show changes over time. An interactive map is possible through geovisualisation.
Geovisualisation (Geographic visualisation) is the analysis of geospatial data with interactive visualisation tools. Geospatial data describes objects linked to a place. E.g. would be the use of a standard format code as the Alpha-3 [ISO, 2022]. The Alpha-3 code 'NLD' links to the name 'Netherlands (the)'. A geovisualisation system links this code to the geographic location of the country. A visual element can be passed on to the linked country. If the output is a map of the world, the Netherlands could be given a different colour to highlight the location. In practice, a variety of information can be used as geospatial data. Python has libraries that allow for working with geospatial data. A few notable python libraries will be discussed in this section.
GeoPandas uses the datatypes from pandas to perform geospatial operations on data sources [developers, 2021]. Geopandas combines the data manipulation of pandas with the manipulation of geometric objects of the shapely python package [Gillies et al., 2013]. This package is based on GEOS. GEOS is a library for computational geometry and the main dependancy for the geospatial operations [GEOS contributors, 2021]. The focus of GEOS is on geographic information systems (GIS). GIS is the most common type of database used for geographic data and is used in many geovisualisation libraries. GIS systems contain software tools for analysing and visualising the data stored in its database.

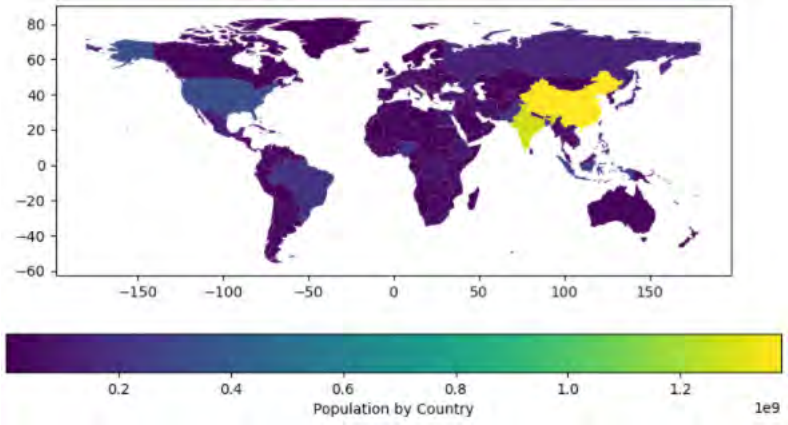



Figure 3: Example of GeoPandas figure. Source: GeoPandas,
*https* : *//geopandas.org/en/stable/docs/user*$_g$*uide/mapping.html*

geoplot uses GeoPandas to create cartographically plotted standard maps [Bilogur, 2022a]. geoplot creates more realistic looking maps through projection. Projection is a common method used in geographic visualisation. Projection maps all the points on a geographic surface into two dimensions

[Bilogur, 2022b]. This creates a flat image of a three dimensional object. The result looks more like the actual geographic place it represents instead of an flattened image.

Figure 4: Example of geoplot figures. Source: geoplot,
*https* : *//residentmario.github.io/geoplot/index.html*

Figure 5: Difference between unprojected an projected map of the United States. The left figure is unprojected. The right figure is projected. Source: *geoplot*,
*https* : *//residentmario.github.io/geoplot/user*$_g$*uide/W orking*$_w$*ith*$_P$*rojections.html*

The graphing library plotly using GeoJSON to allow for geographic visualisation. The geometry data is formatted into the a geojson format. GeoJSON is a standard format for storing geospatial data that differs from the GIS standard. The format is based on the JSON format. The geojson data is used by plotly's graphic drawing arguments to visualise the geographic data.

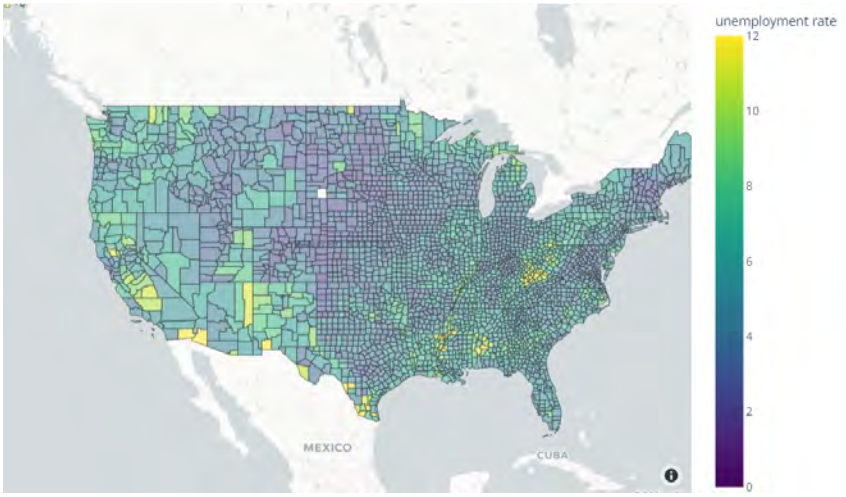
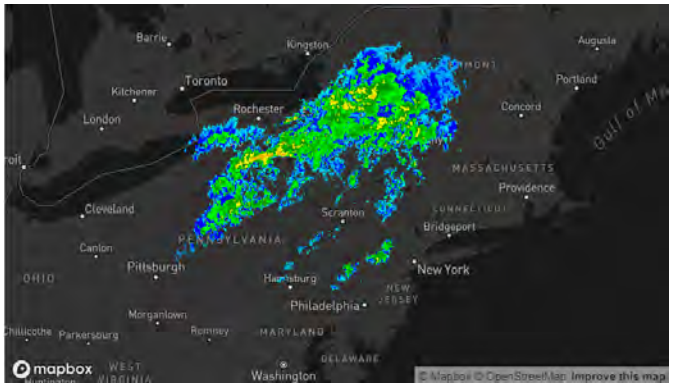



Figure 6: Example map from plotly library. Source: Plotly,
*https* : *//plotly.com/python/mapbox − county − choropleth/*

Some plotly graphs use GeoPandas or the geographical maps from the mapbox enterprise[Mapbox, 2022a]. Mapbox features highly detailed geographical maps of the world and provides the option of building customisable maps through its JavaScript library Mapbox GL JS [Mapbox, 2022b].


(a) Detailed world globe. Source: Mapbox GL JS, *https : //docs.mapbox.com/mapbox − gl − js/example/globe/*



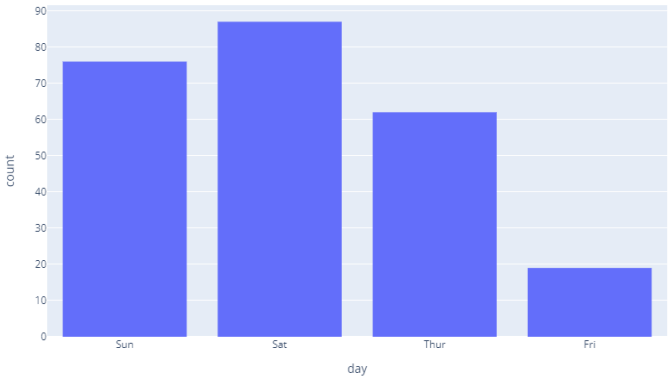

(b) Map with radar weather image. Source: Mapbox GL JS, *https : //docs.mapbox.com/mapbox − gl − js/example/image − on − a − map/*

Figure 7: Mapbox GL JS examples

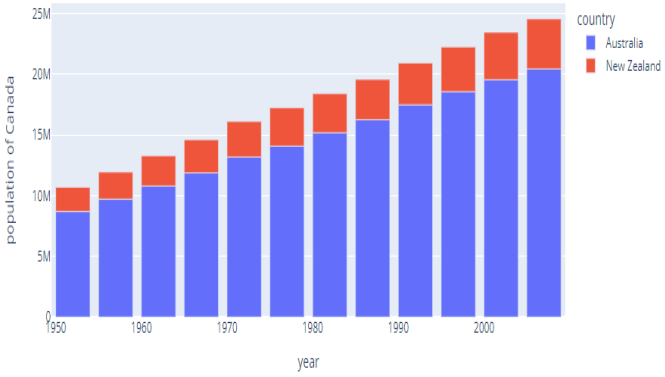
[Evangelidis et al., 2018] mentions multiple 3D geospatial visualisation frameworks based on the WebGL. Many frameworks offer options for animation, (interactive) motion and integrated geospacial functionalaties such as spatial referencing and overlaying. Google Earth is an example of a 3D visualisation of the planet. The Java platform has a 3D API, called Java 3D, that features geographic 3D projects. In their study, [Zhang et al., 2022] found that people experience more comprehension of the data when looking at choropleth maps (heat maps) compared to looking at data tables. These findings correspond with previously mentioned works in that implies people tend to process visual information easier than textual information.

## 5.2 Timeline and Graphs

A timeline is a graphical representation of a time period. In most instances of a timeline, there is a horizontal axis that denotes a period of time with regular time intervals. Elements are placed along the timeline to show their changes or an evolution over the chosen period. Common visualisations of timelines are histograms and bar/line/scatter graphs. These graphs show the elements in every time interval along the timeline to visualise the differences of those elements over time. The bar graph counts the frequency of elements per category. Every bar represents a category in which an element has a frequency. The histogram is a type of bar graph meant to count the frequency of elements over a time period. The bar categories in a histogram are time intervals. Line graphs trace the frequencies of elements over time by drawing a line between individual points. Scatter graphs plot individual points alongside 2 axis. Scatter graphs have the added option for a third dimension in the form of size. The frequency or weight of each plotted individual point can be visualised through the size of the point. A bigger point means a higher frequency or weight.

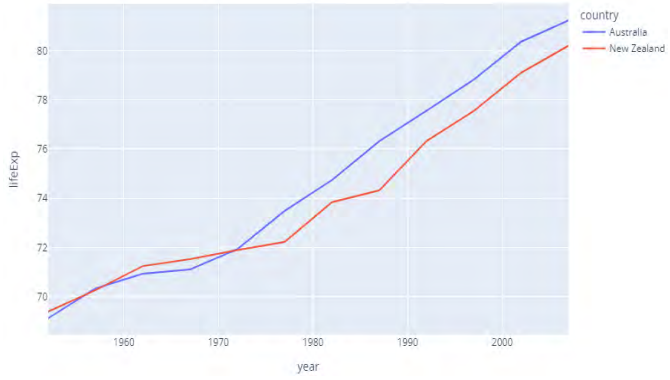



(a) Example of a histogram. Source: Plotly, *https : //plotly.com/python/histograms/*

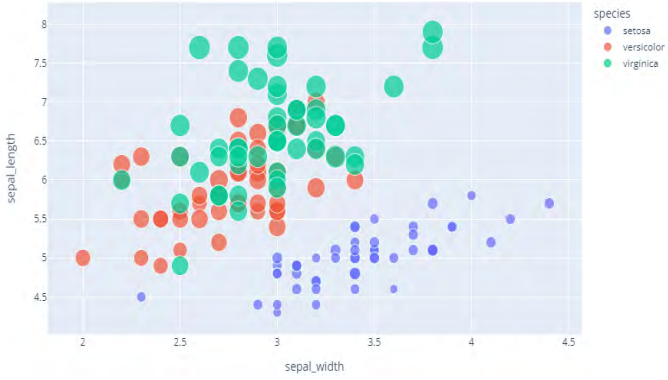



(b) Example of a bar graph. Source: Plotly, *https : //plotly.com/python/bar − charts/*

Figure 8: Histogram and bar graph




(a) Example of a line graph. Source: Plotly, *https : //plotly.com/python/line − charts/*

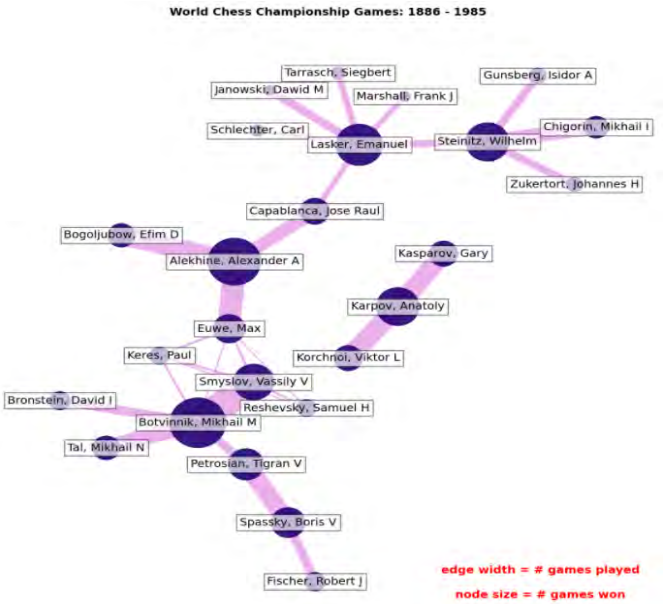



(b) Example of a scatter graph. Source: Plotly, *https : //plotly.com/python/line−and−scatter/*

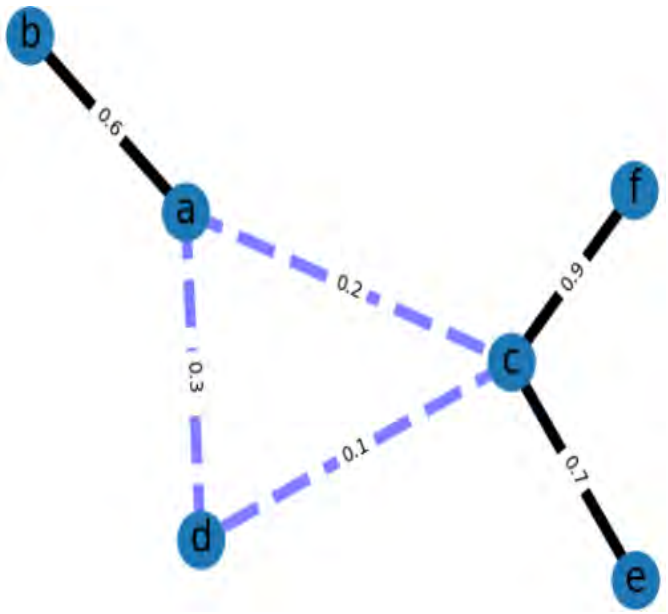
Figure 9: Line- and scatter graph

## 5.3 Network diagram

Network diagrams visualise the relationship between objects through a collection of nodes (vertices) and lines (edges). Every object in the diagram is denoted by an individual node. Two nodes are linked by a line when they share a relationship with each other. Nodes can bring additional meaning through visualisation by their shape, size or colour. Colours, size or shapes can be used to represent the weight of a node or the category a node belongs to. Furthermore, nodes can poses names or other information that is stored around or inside the node in the form of text or images. This is called a labeled graph. The lines between the nodes can contain information to detail the relationship between two nodes. E.g. A line with a numeric value of 100 could mean there is a distance of 100 units between two nodes, or that there have happened 100 occurrences between the nodes. This is called a weighted graph.




(a) Example of a labeled network diagram. Source: NetworkX, *https://networkx.org/documentation/stable/auto_examples/drawing/plot_chess_masters.html#sphx-glr-auto-examples-drawing-plot-chess-masters-py*




(b) Example of a weighted network diagram. Source: NetworkX, *https://networkx.org/documentation/stable/auto_examples/drawing/plot_weighted_graph.html#sphx-glr-auto-examples-drawing-plot-weighted-graph-py*

Figure 10: Labeled- and weighted network diagrams

Besides showing the relationship between objects, network diagrams can be used to show the depth of an object. An example would be a family tree. A family tree can be visualised by a network diagram in the form of a tree graph. Every row in the graph consists of people from the same generation. Horizontal lines represent sibling relationships whereas vertical lines represent children and predecessors. The family tree is an example of a network diagram showing the depth (number of family members and generational span) of an object (family).

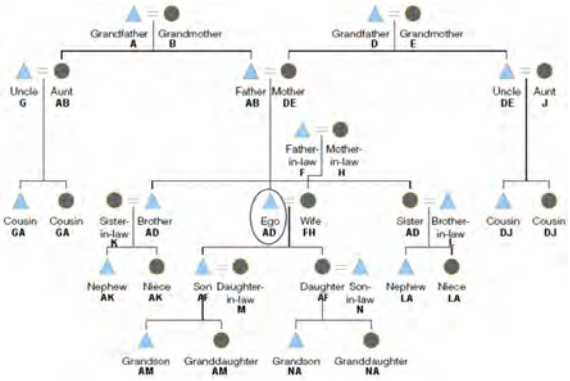
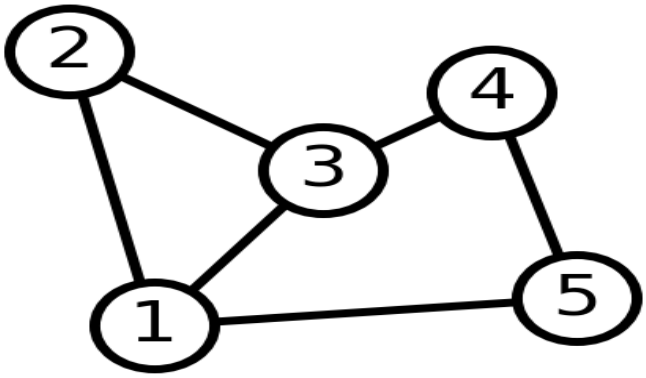
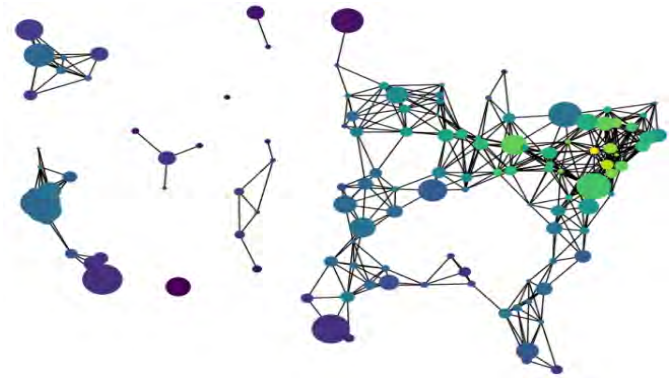



Figure 11: Hypothetical family tree diagram. Source: Luis Alan Navarro-Navarro, *https : //www.researchgate.net/figure/Hypothetical − family − tree − diagram − with − surnames − as − capital − letters − and − family*$_{fig}9_3$24680378

Network diagrams can be very simple when only a few objects are considered. The diagrams quickly become more complex when there are many objects, categories and relationships between objects.




(a) Example of a simple network diagram. Source: NetworkX, *https : //networkx.org/documentation/stable /auto*$_e$*xamples/basic/plot*$_s$*imple*$_g$*raph.html#sphx*$\_$*glr − auto − examples − basic − plot − simple − graph − py*



(b) Example of a complex network diagram. Source: NetworkX, *https : //networkx.org/documentation/stable/auto*$_e$*xampl glr − auto − examples − drawing − plot − knuth − miles − py*

Figure 12: Simple- and complex network diagrams

The python package NetworkX allows for the creation of complex and dynamic network diagrams [Hagberg et al., 2008]. Libraries like Plotly use NetworkX and other graphing modules such as igraph [igraph core team, 2022] for the generation of the network diagrams.

## 5.4 Data table

A direct way of visualising the data is to display the data in a table. Queried data can be displayed in a tabular format to offer an intuitive overview for the user as most people are familiar with tables. Rows in the tables could reflect the historical entities with the columns detailing the entries characteristics. Data tables offer multiple forms of customisability and interactivity. The amount of historical entries that are shown can influenced by the filters placed upon the table. Users could fill in requirements for parameters to find entries that fit the characteristics that they are looking for. Which columns are used and shown in the table can also be changed to the users liking. Furthermore, interactivity can be added to the table by allowing users to alter the table while looking at them. Removing, filtering and selecting rows and columns are forms of interactivity that are supported by tables. Selecting certain rows could be used to alter other forms of visualisation. E.g. selecting a country in a data table filled with countries creates an information box with detailed information about the selected country in the form of text or graphs. An all-encompassing example of an interactive data table is shown in figure 13. It features filtering, deleting and selecting options. Specifically, selecting countries makes them highlighted in the bar plot below the table.

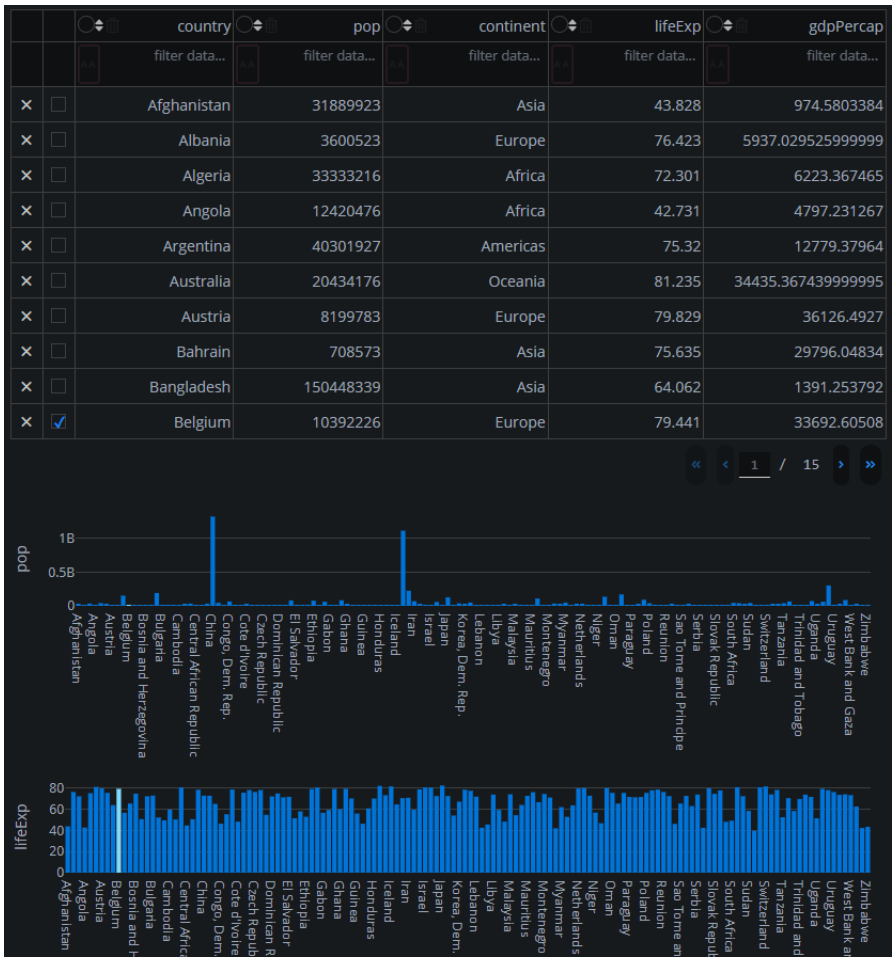



Figure 13: Interactive DataTable. Source: Plotly Dash, *https : //dash.plotly.com/datatable/interactivity*

## 5.5 Word cloud

Word clouds are visualisations of text data. A figure consisting of tags is created. Generally, every tag is a single word or phrase representing an object. Word clouds can vary in shape. Sometimes, the shape of the cloud itself conveys a message from the data. The size of the individual tags usually represent the frequency or significance of the words. Other aesthetic characteristics like colour or font can be used to differentiate the tags or show tags that share a category. The tags are created by locating individual words or phrases inside a text data. After the frequency of every tag is calculated, the tags are put together in a figure in the required shape. Word clouds can be used to emphasise the most common words inside a data set. This method, called keyword summaries, is useful to attract users as it is more visually striking than other visualisation methods. Another use for the word clouds is focusing users on specific parts of the data. Research done by Lohmann et al. has identified what characteristics of the word cloud grabs the focus of users [Lohmann, S. and Ziegler, J. and Tetzlaff, L, 2009]:

- Size
  Larger tags attract more attention and are found quicker than smaller tags
- Position
  Tags in the center of the cloud attract more attention than tags around the sides of the cloud. Tags in the upper left quadrant were found and remembered more than tags in the other quadrants.
- Layout
  The ease of finding a tag is dependant on its layout. Finding a specific tag is most easy with a sequential layout with alphabetical sorting. A circular layout with decreasing popularity.
- Performance
  Word clouds are not effective means of searching for specific words. Furthermore, user engagement increased with word clouds they found more fun and aesthetically pleasing, even when the clouds gave worse performance.

These findings by Lohmann et al. can be used to aim attention towards specific elements of a visualisation. This is not a functionality that would be useful for historians using the visualisation as a tool to do historical research. It can be used to draw the attention of a general public to information in the data that might be of interest to them.
Figure 15 shows a challenge that is present when creating word clouds. When phrases consisting of multiple words are used, the software needs to be told what the complete phrases are and what single words are phrases by themselves. The figure contains words that are part of phrases and not complete terms by themselves.

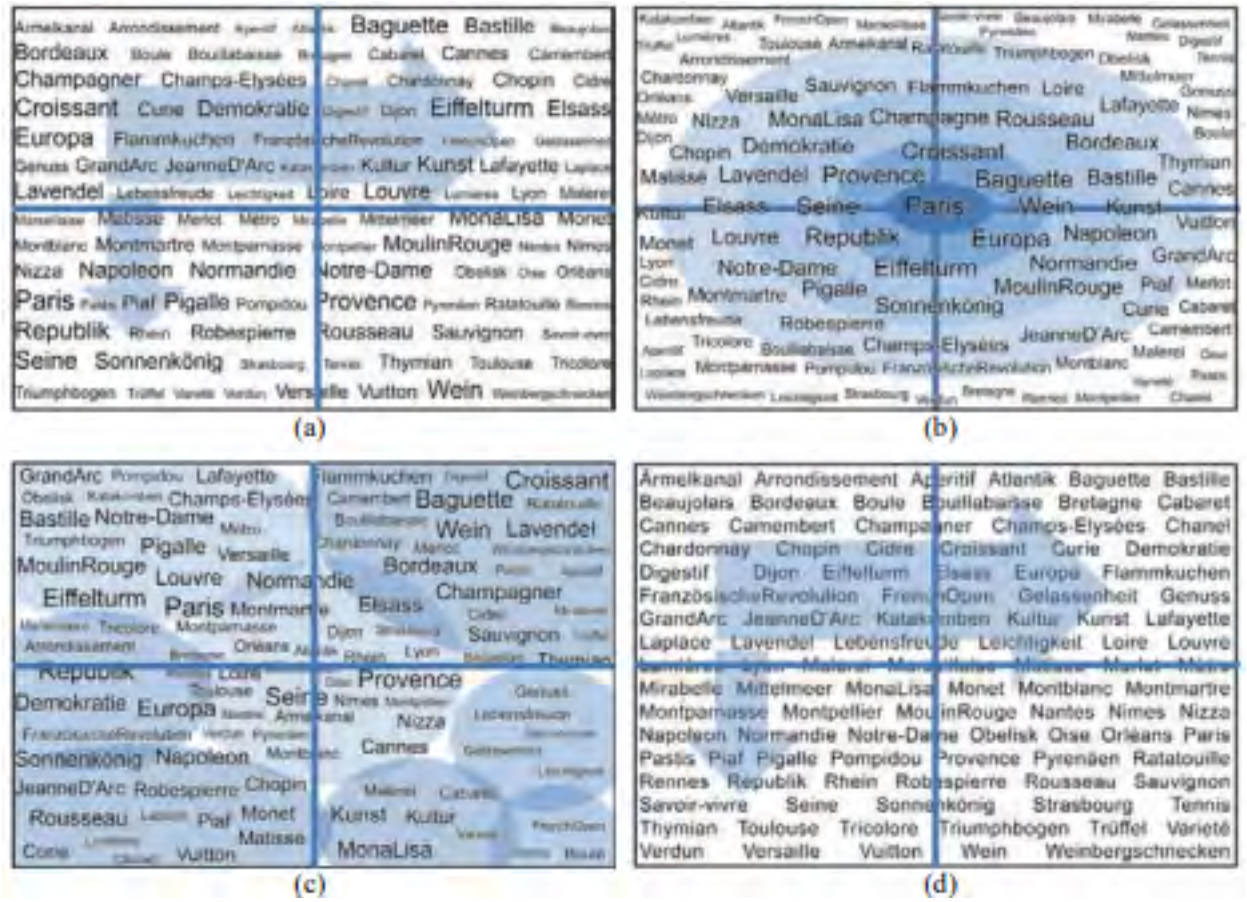



Figure 14: Word clouds used for the tag cloud performance research. (a) sequential (alphabetical sorting), (b) circular (decreasing popularity), (c) clustered (thematic clusters), (d) reference (sequential, alphabetical sorting, no weighting of tags) Source: Lohmann et al.


Figure 15: Example of a word cloud showing frequency of subject areas from the professor data set.

# 6 Assessment of visualisation software

This section describes three possible software tools to be used for the creation of the final product. Every tools has been assessed for the available visualisation methods and interactivity options.

## 6.1 Selection requirements for visualisation software

The following requirements were used for the selection of the software:

- Offers support for the chosen programming language
- Offers features for the specific visualisation methods that were chosen in the research questions 4
- Offers extra features that could be implemented in the future in the case more visualisation methods are needed
- Scalability, the software should support a big data set. The tool should be usable in the case of possible future expansions of the data set
- The software should be supported by the developers

The NodeGoat software was introduced as an option at the start of the project. After the collection of some visualisation software requirements, the visualisation options of NodeGoat seemed to fit these requirements. No programming is required when using the tool. Therefore, the programming language requirement was not relevant for the selection of NodeGoat.
At the beginning of the project, it was established that the primary programming language would be Python. Through previous programming work, JupyterLab was known as a tool that can be used for the quick creation of pieces of Python code. The software fits the use of Pandas, the data analysis tool, and Plotly, the Python graphing library. One of the earlier ideas within the projects was to use a web framework, E.g. Django, to host the visualisations. JupyterLab could be used to process the data and create individual graphs. The graphs could then be imported into the framework.
The option of Plotly Dash came up after researching the visualisation possibilities of the Plotly library. The Plotly documentation mentioned Dash frequently as an option for data visualisation. The dashboard framework could be a good fit for the project as it focused on the visualisation of data on an easy to deploy web based model. Other dashboard software was also an option. But, ultimately, Plotly Dash was chosen as dashboard option because it features a heavy integration with the rest of the Plotly library and is in active development. More features were added to Dash even during the project's lifetime.

## 6.2 NodeGoat

NodeGoat is a web-based program for data handling, analysis and visualisation [Bree, 2013]. NodeGoat can be used to collaboratively work on projects and share them with others. The program allows users to create, link or import their data to the NodeGoat environment and visualise the data through one of its premade visualisations. NodeGoat has paid services that allow users to

work with APIs and gives them access to a custom public front-end to showcase their project's visualisations.

Working with NodeGoat requires no programming work and can all be done within the web browser. After setting up a project, users create objects and fill them out with attributes. Each attribute can be a reference to an attribute of another object, or a category. Categories can be seen as labels. Labels can be used to show the similarities and differences between objects. The categories all fall under a classification. Classifications are entities that function as lists for a group of categories of the same type. E.g. an object named person can have an attribute religion. A classification called religion can be made with a number of categories that fall under the classification, e.g. Christianity, Islam, Judaism. The attribute inside the person object references the religion classification. When a person is added as a person object with the Christianity as its religion object, the attribute is linked to the Christianity category in the religion classification. Another feature of the object is the ability to create sub-objects. Sub-objects are objects within an object. Sub-objects are used to store changing/contextualised attributes from the object. This is done by using time references in the from of a data range or begin- and end points, and geographical references. The sub-object can also have its own attributes and references. An example for a sub-object is a person's birth. If a person is modeled into a person object, a sub-object for their birth can be created to contextualised the time and place of said birth. This means that a date and location can be set in the sub-object. NodeGoat features a list of cities that can be referenced in the location. In the case of the birth example, visualising the birth means that a dot will appear on the referenced location (e.g. a city) on a geographical map. The dot would only appear on or beyond the referenced date the birth took place on.
Sub-objects differ from classifications because they are part of the object, and contextualise an attribute of that object through time- and space references and its own attributes. Classifications on the other hand, feature a list of categories that are essentially labels that objects can be categorised to.
When objects and classifications are created, NodeGoat automatically builds a data model in the background that connects all elements. When all objects are created, data entries can be added to the project. NodeGoat constructs a database by itself when data is added. Data can be added through manual addition of object entries, through importing csv files, or linking data from other resources through APIs, SPARQL queries, static links or JavaScript scripts. Added objects can be viewed, edited and deleted in the data page.

Visualisations can be constructed after data entries have been added. There are three visualisation methods:

- Geographical Visualisation. Shows a geographical map with all geo- and time-referenced attributes of the objects. All attributes on the map that belong to the same person are linked by (animated) lines. E.g. a birth dot connects to a graduation dot, which then connects to a death dot. This makes it possible to track the lifelines of individual objects. The integrated timeline can be used to view the evolution of attributes over time. The timeline can also be played backwards. Figure 44 shows an example of the geographical visualisation.

- Social Visualisation. Builds a network diagram objects' attributes with categories. E.g. a group of people, all modeled as individual objects, have a nationality attribute ranging from

Dutch to German to French etc. A classification Nationality consists of all these nationalities. All objects, resembling the group of people, are linked to the categories (nations), that match the people's nationality attribute. Classifications can be switched on and off to expand or shrink the network diagram by including/excluding the classification's categories. Users can manually move parts of the network. The social visualisation also includes the same integrated timeline as the geographical visualisation. Figure 45 shows an example of a network diagram with multiple attributes linked whereas figure 46 shows a diagram with only one attribute active.

- Chronological Visualisation. Displays a timeline through a bar graph. Every time-referenced attribute is stacked in bars per year to give a visual overview of the amount of objects per year. The integrated timeline, once again, functions the same as in the other visualisation methods. 47 shows an example of the chronological visualisation.

Users can create scenarios that save all visualisation settings made previously. Filters are used to select a subgroup of the objects to be visualised, e.g. all professors or all students. The visualisation scope can be changed to focus on certain time periods, geographical locations or object attributes. Attributes can be cross-referenced. The cross-referencing feature is used for the social visualisation to change the structure of the network diagram. Visual elements can be changed, e.g. changing dot, line and map properties.
NodeGoat has the following advantages and disadvantages:
Advantages:

- Project is stored online and accessible by all project members
- NodeGoat is easy to use and features documentation on every function it has
- The project structure and database are configured and updated automatically
- Allows for importing of data from files, urls or APIs
- Features pre-made high quality visualisations
- New features are constantly added by the developers as a result of commissions by research projects

Disadvantages:

- The variety of visualisations is limited to three types
- Importing, changing and visualising data takes a long time
- The amount of data that can be viewed at once within NodeGoat is limited because it has a maximum memory of 128 megabyte. This limits the amount of objects that can be viewed at once in larger data sets.
- The amount of data that can be visualised at once within NodeGoat is limited

## 6.3 JupyterLab

Jupyter was previously mentioned in section 2.2. The Jupyter project is a non-profit open-source program that can be used in the fields of interactive data science and scientific computing [Jupyter, 2022b]. Project Jupyter supports over 40 programming languages [Jupyter, 2022a]. There are currently two versions available for use: The original Jupyter Notebook, and the newly released JupyterLab. Both versions serve as document workspaces where users can create and share notebooks (documents). Notebooks functions as documents. Users can create cells in the document that each can be executed separately. Information executed in the cell can be used throughout the whole notebook and gets updated with every cell execution. Single cell execution helps real time development as users can write a piece of code and run it immediately. This decreases runtime as only a small part of the entire code in the notebook has to be executed. An example of a cell and its output are shown in figure 48.
JupyterLab has the following advantages and disadvantages:
Advantages:

- Files can be shared between project members
- Project members can collaborate on files simultaneously
- Code can be run in parts with low runtime
- Supports many programming languages and interactive outputs

Disadvantages:

- Requires users to have programming knowledge
- The cell based coding approach of JupyterLab is likely not intuitive for all people
- Code in the Notebooks need to be exported to other frameworks in order to be published publicly.
- JupyterLab needs to be locally ran in order to use the server

## 6.4 Plotly Dash

Plotly Dash is a low-code framework, built upon Plotly.js and React.js, by Plotly [Plotly, 2022d]. The Dash Enterprise Platform allows users to build business oriented Dash apps that are scalable and feature multiple IT related services such as authentication [Plotly, 2022a]. Dash has an open source option to create their own custom data dashboard. Dash focuses on working with data and lets users build interactive data apps in short time. The framework supports Python, R, Julia and F. The Dash apps are rendered in the web browser and can be integrated in other frameworks.

Dash works with Dash Core Components, Dash HTML Components and Dash Callbacks. The Dash Core Components module includes components for interactive user interfaces. Examples are dropdown bars and sliders that allow users of the app to interact with the graphs. The graphs themselves are also constructed as a core component. HTML components make up the other parts of

the app. The HMTL components shape the lay-out of the app through HTML, CSS and JavaScript components. E.g. HTML div components can be used to structure the app comparable to how a html-based website would be made. Dash uses callbacks functions to automatically detect when Dash or HMTL input components change value. The callback then updates the designated output component. This callback system allows users of the app to interact with the dashboard. Dash has high compatibility with the Plotly library. Plotly graphs can be made in Dash to populate the app with visualisaions. Dash also includes its own DataTable component and other advanced forms of visualisation.
Plotly Dash has the following advantages and disadvantages:
Advantages:

- High customisability for the creation of data dashboards
- Integratable in other frameworks
- Dash documentation describes all of its components. Furthermore, a lot of documentation exists online.
- Can be upgraded to the Dash Enterprise service for scalability and IT features
- Dash has the option of hot-reloading, which means that the web browser will automatically refresh when the code is changed.
- Plotly Dash is in active development and new features are continuously added

Disadvantages:

- Requires more knowledge and effort than a platform like NodeGoat
- The hot-reloading feature takes a bit of time and sometimes breaks which makes the process of implementing and checking new features longer and more arduous.
- The dashboard is scalable for more data, but it takes increasingly longer to load elements on the page if more data is used. This can be circumvented by scaling down the amount of information that is requested at once.

## 6.5 Conclusion

It was decided that both a NodeGoat and a Plotly Dash prototype would be made. After the discovery of Dash, it was favoured over JupyterLab as it didn't need a separate framework to be deployed. It's integration with the Plotly graphing library and focus on building data driven visualisations made it a better choice for a model than JupyterLab. NodeGoat was still pursued as it had the interest of the project group. If featured a different route of no-programming data visualisation compared to Plotly Dash's building from the ground up approach.

# 7 Implementing two interactive visualisation environments

This section covers the creation of two data visualisation models in two different environments. The first model is configured in NodeGoat and features three visualisation types. The second model is a dashboard built in a Plotly Dash app written in python code.

## 7.1 Programming Language

The visualisation is done within the Python programming language. A majority of the code written is part of the Plotly library. Specificaly two Plotlty libraries: Plotly Dash, covered in section 6.4, and Plotly Express. This section will elaborate more on the plotly.express module.

The Express module is a built-in part of the plotly library [Plotly, 2022e]. Before Plotly Express, graphs where made through Plotly Graph Objects. The Graph Objects in the plotly.graphobjects module consist of python classes. The main class is the Figure class which configures the entire figure. The contents of the figure is then further build up through adding trace objects and layout update objects. Users would write these objects one for one in their code. Plotly Express simplifies this process by creating the entire figure at once in a single line of code. Plotly Express can make the same figures that Graph Objects can, but does it with 5 to 100 times less code [Plotly, 2022e]. The code difference between Plotly Express and Graph Objects is displayed in figure 49. Currently, Plotly Express features more than 30 different type of figures.

Plotly recommends the use of Plotly Express over Graph Objects [Plotly, 2022c]. Graph Objects gets recommended in a few cases:

- Some Plotly figures are not yet available in Plotly Express. Graph Objects are needed to create these figures.
- Some Plotly figures would take more effort to create with Plotly Express as opposed to Graph Objects. This is mostly the case for figures with different types subplots, axis and different types of traces. Using Plotly Express forces the user to make many alterations which increases the code length and feels like a roundabout way of constructing a figure. Graph Objects can be used to start with an empty figure and add traces and update attributes one by one [Plotly, 2022c].

The Plotly libraries handle the visualisation of the data. Before visualising, the data has to be converted into a data structure that the Plotly code can work with. The data from the excel files is therefore processed into Pandas DataFrames. Pandas is a library suited for data structures and data analysis within the Python language [Pandas, 2022a]. The data structure used by Pandas is the Pandas DataFrame. A dataframe is a 2-dimensional data structure, similar to a table or an array. The dataframe is able to store many different types of data including both numerical and categorical data [Pandas, 2022b]. When a column in a dataframe is selected, the data is converted into a Series object. A series is an one-dimensional array that stores all data values from a column. The dataframe is the data structure that stores the data in a convenient matter. The series data structures are then used to find specific information needed for the visualisation.

## 7.2 NodeGoat model

A NodeGoat model for the LUCD project has been made with the excel files featuring Professor, Student and rectores magnifici data. The model consists of one object and 10 classifications. The object in the model is 'Person'. All professor, student and rector entries are added as persons. The distinction is made through the classification 'Type of person'. The following sections show the model parts in detail.

### 7.2.1 Person object

The object in the model is the Person object. Every person in the Professor, Student and rector are converted into a Person object. The person object has 17 attributes and 8 sub-objects. All excel files featured different attributes. This is reflected in the object attributes as not every attribute is applicable to all person entries. The attributes are listed in table 11, together with their type and to which data set it is relevant to.
The sub-objects all consist of events in the lives of the people in the data sets that contain forms of geographical- and/or time-referenced information. The sub-objects are listed in table 12.

### 7.2.2 Person classifications

Person classifications can be used to create a network diagram and connect persons with the same attributes. The characteristics in the data sets that can be used for this purpose are translated into categories. These categories are linked to the Person object. Every classification has a number of categories that are present in the data. The classifications that are featured in the model are listed in table 13.

### 7.2.3 Complete model

A feature of NodeGoat is the automatic data structure that is created when user add objects and classifications. The structure of the NodeGoat model is shown in figure 50. The main body is the Person model with 51.488 objects. The 10 classifications are linked to the referenced attributes in the Person object. Two separate objects are part of the model: City and Geometry. These objects are part of NodeGoat's default database and provide the data for geographical references. 4 visualisations of the professor data are shown in figures 44 through 46.

## 7.3 Plotly Dash model

Plotly Dash was used to create a data dashboard. The dashboard is a Python written application that functions as a website once deployed. This section covers the design details of the dashboard. A detailed user guide is included in the appendix A. The guide covers how to use the dashboard and explains all the elements of the dashboard.

### 7.3.1 Dashboard structure

The dashboard that visualises data in four distinct sections for every data set. The dashboard is structured in multiple pages and sub pages. Users can navigate the pages through two navigation bars One navigation bar is used to navigate through the data sets, home and source pages. When on a page about a data set, the second navigation bar is used to navigate between the four visualisation pages. There are a few reasons why there will only be 1 visualisation component showed per page:

- The load time per page improves significantly. The visualisation components load up quickly when switching between the pages. This gives the dashboard a more dynamic and responsive feeling and allows the user to interact with the contents immediately.
- The amount of visual elements on the page is limited which ensures viewers are not overwhelmed. Less elements on screen, especially when they are visual of nature, allows viewers to focus on content that is on screen. This makes the visualisation seem easier to interpret and understand.
- There is minimal scrolling involved. Most of the contents can be seen all at once. This reduces the amount of time users spend on looking for what they want.

### 7.3.2 Data structures for responsive visualisations

The dashboard uses data that is passed into dataframes. As the visualisations use different data, multiple dataframes have been made. Every dataframe consists of one or more characteristics (E.g. name, birth city, religion, enrollment year). There are two types of dataframes created for the dashboard. The first type is a subset of the main data sets where the data originated from. The subsets contain the same amount of rows (I.e. the entries or people). The amount of columns differs, depending on which characteristics where needed for the visualisation. This type of dataframe is used for the data tables and functions as a small database that can be used to query information into the data tables. The second type of dataframes breaks down a single characteristics in the data set. In the dashboard, the characteristics are named subjects. Every subject describes a characteristic of the people in the data. For the sake of consistency with the dashboard, from this point on we refer to the characteristics as subjects. They mean the same thing, that is, the characteristics as explored in the data section 3. A subject dataframe is constructed as follows: the entire column in the original data set is taken, as well as the column containing the relevant date. This is done in order to create time based visualisations. E.g. the student data describes student enrollments. Every entry in the dataset is an enrollment and features a date of enrollment. The visualisations are year based, and thus only the year is required. In total, the year is taken together with the subject the dataframe focuses on. E.g. we look at the subject world regions, the region where students come from. For every year, the unique region values are counted. This results in a dataframe with

a column of values, a column with the value count, and a column detailing the year. In the case of the student data, both a column for year, and a column for century are included in the dataframe. The century data was included in the student data set. For the professor and rector data set, the century was calculated from the year while processing the data into a dataframe.
An example of a subject dataframe is shown in figure 52. In this figure, the subject is 'region' which reflects the world regions where newly enrolled students came from. For all years in the database (the years 1575 to 1812) the world region information is broken down. The enrollments are counted per individual year, century, and region value in order to create time-based visualisations that can focus on multiple time scopes. The value counts are used for timeline, graph and geographical visualisations.
A subjects dataframe can be broken down further when only the value count per century is needed. Figure 53 shows the world region data sorted per century instead of per year. When only an overall breakdown per region is required. A dataframe of the total value count per unique value is used. Figure 54 shows this dataframe.

The subjects are passed into dataframes. There are separate dataframes for every subject. Some extra dataframes consisting of multiple subjects are created for more complicated visualisations that need the information from multiple subjects. Dataframes are kept as small as possible in order to shorten processing time as only a small part of all data has to be queried.

### 7.3.3 Visualisation sections

All three data pages use the same visualisation components. For that reason only the components in the students page will be covered in this thesis as the Professors and Rectores Magnifici pages work functionally the same. The difference in visuals will be because of the different data available between the data sources. The following four visualisation sections are featured:

- Timeline. The timeline features information per subject over the entire time span of the data set. This section includes graphs and data tables.
- Subject information. This section contains more detailed information per subject. The section includes graphs and data tables.
- Geographical information. Gives a geographical representation of the data. Multiple map types and featured, supported by data tables.
- Individual information. Includes a data table where users can search for people in the data set. The search can be filtered for all available subjects.

### 7.3.4 Visualisation interactivity

Every graph in the dashboard offers interactivity options for the users. All graphs are made with Plotly and therefore offer an options bar that is integrated in all Plotly graphs. When a user hovers their mouse over a graph, a toolbar appears. The functions within the toolbar are explained in the user guide A.2.
Other interactivity is included in the form of hover information. While hovering over data points in the graphs, information about the data point is shown. Additionally, more detailed information

concerning the specific data point and overall graph data is updated next to the graph while hovering.
Exporting of data tables and graphs is included so users can save the data they discovered.
All graphs are supported with extra options to change the parameters of the graph. For example, time scopes, subjects and graph types can be changed.

# 8 Discussion

The prototypes provided in section 7.2 and 7.3 provide possible methods of data visualisation of the available historical data. The two found applicable approaches are:

- NodeGoat. A web based program for importing data and visualising data through three detailed visualisation methods. The work that needs to be done to create the visualisations is limited to creating (sub) objects with corresponding attributes an classifications that link objects with the same attributes. NodeGoat automatically creates a database and data structure for the project. Downsides are the loading times when importing, changing and visualising large amounts of data. The amount of objects that can be viewed and visualised is limited which can be problematic for a project with a large amount of data like LUCD.

- Plotly Dash. An interactive, highly customisable dashboard that features many possible options of data visualisation an aesthetic lay-outs. The dashboard can be run on its own on a server or be integrated in other frameworks. The biggest challenges for the Plotly Dash Dashboard are the time required to make the dashboard and the loading time challenges when large amounts of data are passed to the visualisations.

Both methods are applicable to the LUCD project. The available data sets reviewed in section 3 contain information that is relevant to the main perspectives of interest mentioned in 1.1. Through research questions gathered by requirement interviews 4 it was determined that the data can be visualised through the use of timelines, geographical maps and graphs. These methods have been examined in the related works 2 and methods 5 sections. Both Visualisation programs, NodeGoat 6.2 and Plotly Dash 6.4 posses the option to implement the visualisation methods that are applicable to the data. The models shown in this thesis in sections 7.2 and 7.3 are showcases of the possibilities the programs have.

The NodeGoat model includes three commonly seen visualisations; the geographical map, network diagram and timeline. Similarly, the Plotly dashboard model features visualisation methods seen in previous works that have been covered in section 2. The geographical map, timeline and search databases can be found in the dashboard. A big focus of the dashboard are the line/bar/scatter graphs. Many of the characteristic in the data sets can be visualised in the three graph types for a statistical understanding of the data. NodeGoat's timeline is comparable to the year timeline present in the first visualisation page of the Plotly dashboard. The geographical map from NodeGoat is more suited to show the travel of people during their lifetime with its animated line capabilities, whereas Plotly's maps are more static. The NodeGoat model does not allow for all data objects to be visualised at once. This means that complete visualisations of all people can not be shown in one figure. Multiple scenarios have to be made in order to get the complete information of the data. This could be a hindrance for scientific research. The importing of data takes a long time when working with big data sets in NodeGoat.
The dashboard distinguishes itself through the search function where users can find individuals in the database. The feature is similar as seen in the professor search catalog of the University of Leiden 2.3.8. The dashboard adheres to the principles stated by Tufte, as visualisations are data heavy, focused on the data and show variation in data presentation through multiple types of graphs. Interactivity is present in both NodeGoat and the Plotly Dashboard. NodeGoat allows

user to control the timeline in every visualisation, as well as choose which classifications are shown. Users of the dashboard have the option to change parameters of the graphs and choose which characteristics are shown. All visualisations on the dashboard can be saved and exported. A comparison between both prototypes is shown in feature criteria table 14, performance criteria table 15 and user experience criteria table 16. In total, the Plotly Dash prototype is favoured in the majority of the criteria as shown in table 5.

| Protoype | Score |
|---|---|
| NodeGoat | 4 |
| Plotly Dash | 9 |

Table 5: Summary of NodeGoat and Plotly Dash prototype comparison

## 8.1 Limitations

One of the limitations in creating the visualisations was the missing data in some parts of the data sets. Specific geographic data is missing to see where professors and students lived during their time at the university. Furthermore, data regarding their life after their time at the university is absent. For 25 percent of the professors, data regarding their promotion is missing. 37 percent of the employment ending dates are missing. Birth and passing dates are not always complete.
In the student data, information about their passing is missing. 31 percent of students have unknown places of origin. Information about their jobs and religion are also scarce. It seems unlikely that only 7 percent of students were working during their time at the university, and less than 1 percent followed a religion.
The prototypes have not been tested by users as a result of time constraints. Short showings to both the LUCD project group and historians at the Leiden University have taken place. This showings resulted in minor feedback and some requirements that were used in building the prototypes. The final prototypes, as presented in this thesis, have not had any proper feedback by users.

## 8.2 Further Research

The visualisation models have more potential when it comes to data visualisation. Future research can be done into more types of visualisations for both models. Especially Plotly Dash can be researched further to create more intricate visualisation methods. The lay-out of the dashboard can also be changed drastically to the wishes of the LUCD project team. Animation on graphical maps is theoretically possible within Plotly Dash through the use of Mapbox GL JS. An implementation was not able to be made during this project within the dashboard environment. Animated Mapbox maps could increase the depth of geographical data visualisation within the dashboard. The implementation of Mapbox GL JS might be worth further research if the LUCD project team desires more depth in geographical data visualisation. Furthermore, the addition of genealogical information in the dashboard has yet to implemented. But first builds of visualisation methods of family trees have been made, but were not represented in this thesis. Lab1101, the creators of NodeGoat, have shown interest to the project team in implementing different types of visualisation. A collaboration between Lab1101 can be made if the project team creates their own visualisation methods in JavaScript that can be integrated in a specific NodeGoat model. This would greatly

increase the variety and potential of NodeGoat's capabilities as a data visualisation tool. This means that visualisations made with Plotly can be rewritten in JavaScript code and implemented in NodeGoat. Plotly Dash could then be used to create and test visualisations that ultimately would end up in a NodeGoat environment.

Building on the last point. Plotly Dash could be used by digital humanities and data science faculties within Leiden University. Students and staff members could use Plotly Dash as an experimentation platform where new visualisation methods for historical data are developed.

More research has to be done into the acquisition of data regarding the city Leiden and the relation with the university. As mentioned in the limitations, much time and geographic related data is missing or absent in the current data sets to make specific visualisations that can answer the main perspectives of the LUCD project. Many sources for this data already exists as was shown in the section 2.3. Data sharing collaborations between other organisations, e.g. Erfgoed Leiden, and the LUCD team can greatly increase the visualisation options and help historians to get an insight in the history of city and university.

The built prototypes have to be tested by a group of users. Separate test for historians and general audience members could yield more feedback that can be formed into requirements for future improvements of the model. The test can also help decide which of the two prototypes would be a better fit for the LUCD project.

# 9 Conclusion

The main research question this thesis examined was:

*What are applicable software frameworks for the creation of interactive visualisations of a large historical data set about the scholars and students of Leiden University in the past 450 years.*

Two promising data visualisation prototypes were created for the LUCD project. Firstly, a prototype created within NodeGoat that featured easy of use and detailed visualisations with limited visualisation types and slow data processing. And secondly, a Plotly Dash Dashboard that possessed high customisability and variation against greater programming challenges. Both models featured interactivity for users. The dashboard has many interactive options regarding its graphs in the form of customisabilty of parameters.

Visualisation methods were found to research mobility, geographical segregation and social integration problems. Methods that fit these issues are geographical maps, graphs and timelines, Research by historians can be done on historical data sets in order to answer these questions.

The current prototypes are not sufficient to perform meaningful historic research. Too much data is missing or absent in the currently available data sets. Both feature technical limitations. For NodeGoat, there is a limit on the amount of object visualisation. Furthermore, importing data takes a long time with big data sets.

Both prototypes can be improved with further researched. More visualisation methods can be created and implemented in the two prototypes. Collaboration with other historical institutions can be performed to acquire more data that can be used to further research the relation between city and university in Leiden.

# References


[rec, 2022] (2022). Lijst van rectores magnifici van de universiteit leiden.

[Ariadne Schmidt and Wessel Kraaij, 2021] Ariadne Schmidt and Wessel Kraaij (2021). Linking university, city and diversity.

[Bilogur, 2022a] Bilogur, A. (2022a). geoplot.

[Bilogur, 2022b] Bilogur, A. (2022b). Working with projections.

[Blom, 2008] Blom, J.C.H., v. M. R. S. C. (2008). Historische canon van leiden.

[Bree, 2013] Bree, P. van, K. G. (2013). nodegoat: a web-based data management, network analysis visualisation environment.

[CBG, 2022] CBG (2022). Wiewaswie.

[Delpher, 2022] Delpher (2022). Wat is delpher.

[developers, 2021] developers, G. (2021). Geopandas.

[Diblen et al., 2019] Diblen, F., Attema, J., Bakhshi, R., Caron, S., Hendriks, L., and Stienen, B. (2019). spot: Open source framework for scientific data repository and interactive visualization. *SoftwareX*, 9:328–331.

[Enterprise, 2022] Enterprise, D. (2022). Plotly dash.

[Evangelidis et al., 2018] Evangelidis, K., Papadopoulos, T., Papatheodorou, K., Mastorokostas, P., and Hilas, C. (2018). 3d geospatial visualizations: Animation and motion effects on spatial objects. *Computers geosciences*, 111:200–212.

[Faber, 2015] Faber, W. (2015). Newspaper image count (2015), kb lab: The hague.

[Faber, 2021] Faber, W. (2021). Timelapse. kb lab: The hague.

[FactGrid, 2022] FactGrid (2022). Factgrid.

[GEOS contributors, 2021] GEOS contributors (2021). *GEOS coordinate transformation software library*. Open Source Geospatial Foundation.

[Hagberg et al., 2008] Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In Varoquaux, G., Vaught, T., and Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA.

[igraph core team, 2022] igraph core team, T. (2022). igraph - the network analysis package.

[ISO, 2022] ISO (2022). Iso online browsing platform for country codes.

[Jupyter, 2022a] Jupyter (2022a). Jupyter.

[Jupyter, 2022b] Jupyter (2022b). Jupyter.

[Koch et al., 2006] Koch, K., McLean, J., Segev, R., Freed, M. A., Berry, M. J., Balasubramanian, V., and Sterling, P. (2006). How much the eye tells the brain. *Current biology*, 16(14):1428–1434.

[Lab, 2022] Lab, K. (2022). About us kb lab.

[leiden Omstreken, 2008] leiden Omstreken, E. (2008). Historisch leiden in kaart.

[Lohmann, S. and Ziegler, J. and Tetzlaff, L, 2009] Lohmann, S. and Ziegler, J. and Tetzlaff, L (2009). Comparison of tag cloud layouts: Task-related performance and visual exploration.

[Lonij, 2017] Lonij, J., W. M. (2017). Siamese. kb lab: The hague.

[Mapbox, 2022a] Mapbox (2022a). Mapbox.

[Mapbox, 2022b] Mapbox (2022b). Mapbox gl js.

[Ocean, 2022] Ocean, C. (2022). Code ocean.

[Pandas, 2022a] Pandas (2022a). Pandas documentation.

[Pandas, 2022b] Pandas (2022b). What kind of data does pandas handle?

[Pastore, 2020] Pastore, A. (2020). Elevating your multichannel marketing strategy with ai. *WWD*, pages 18–18.

[Perkel, 2018] Perkel, J. M. (2018). Data visualization tools drive interactivity and reproducibility in online publishing. *Nature (London)*, 554(7690):133–134.

[Plotly, 2022a] Plotly (2022a). Dash enterprise.

[Plotly, 2022b] Plotly (2022b). Datatable filtering.

[Plotly, 2022c] Plotly (2022c). Graph objects in python.

[Plotly, 2022d] Plotly (2022d). Introduction to dash.

[Plotly, 2022e] Plotly (2022e). Plotly express in python.

[Gillies et al., 2013] Gillies et al. (2013). Shapely.

[Universiteit Leiden, 2021] Universiteit Leiden (2021). Leiden2022: European city of science.

[Schmidt, 2021] Schmidt, A. (2021). Diverse beelden van de universitaire geschiedenis.

[Tufte, 1983] Tufte, E. R. (1983). *The visual display of quantitative information.* Graphics Press, Cheshire, Conn.

[University, 2022] University, L. (2022). Ontdek de collectie.

[van Nederland, 2022] van Nederland, C. (2022). Canon van nederland.

[Wildschut, 2017] Wildschut, P., F. W. (2017). Narralyzer. kb lab: The hague.

[Zhang et al., 2022] Zhang, J., Wang, Y., Wanta, W., Zheng, Q., and Wang, X. (2022). Reactions to geographic data visualization of infectious disease outbreaks: an experiment on the effectiveness of data presentation format and past occurrence information. *Public health (London)*, 202:106–112.

[Zoeteman, 2011] Zoeteman, M. (2011). De studentenpopulatie van de leidse universiteit, 1575 -1812. 'een volk op zyn siams gekleet eenige mylen van den haag woonende.

# Appendix

# A Plotly Dash User Guide

This section gives a detailed explanation about all elements present in the Plotly Dash dashboard.

## A.1 Dashboard structure

The dashboard consists of multiple pages and subpages. Users can navigate the pages through the navigation bar at the top. There are 5 main pages:

- The homepage
- 3 pages, corresponding to the data sources: Professors, Students and rectores magnifici
- A sources page where the sources of the project are listed

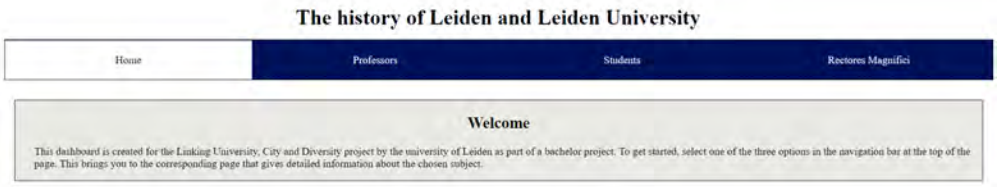



Figure 16: Dashboard homepage

Navigating to one of the 3 data pages brings up the first out of four visualisation pages. A second navigation bar appears under the main bar. The second bar is used to navigate between the four visualisation pages. Every page features one visualisation component. The four visualisation pages will be explained in detail in the sections A.3 through A.6.

## A.2 Visualisation toolbar functionalities

The integrated interactivity tools of Plotly graphs offer a number of functionalities to users. When a user hovers their mouse over a graph, the following toolbar will appear on the top right part of the graph:
Clicking on one of the options in the toolbar gives the following results:

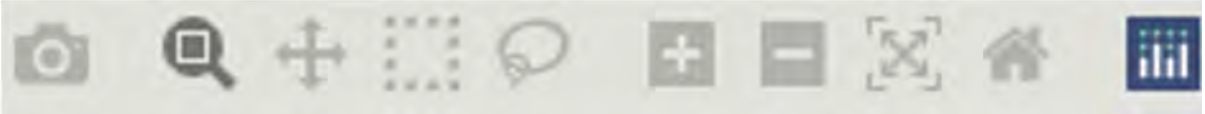


Figure 17: First visualisation page of submenu Students

 The graph will be downloaded and saved as a .png file

 This option is selected by default. The user can zoom into a specific part of the graph by selecting it.

 Gives the user the option to move withing the graph

 Select a part of the graph in the shape of a box.

 Select a part of the graph in the shape of a circle.

 The graph will be zoomed in.

 The graph will be zoomed out.

 The graph will be scaled automatically to show the full graph.

 The axes will be reset and the graph will return to the default settings.

 Link that takes the user to the Plotly website.

Another form of interactivity is the hover sensitivity that is featured in every plotly graph. Hovering over a point in any graph will reveal a block of information on that point.

## A.3 Timeline

The timeline is the first visual component that the users of the dashboard will see. The timeline shows the changes of a subject over the entire available time span. The timelines are supported with light textual information.
The timeline consists of four parts (figure 18):

- Year timeline
- Century timeline
- Timeline settings
- Timeline information

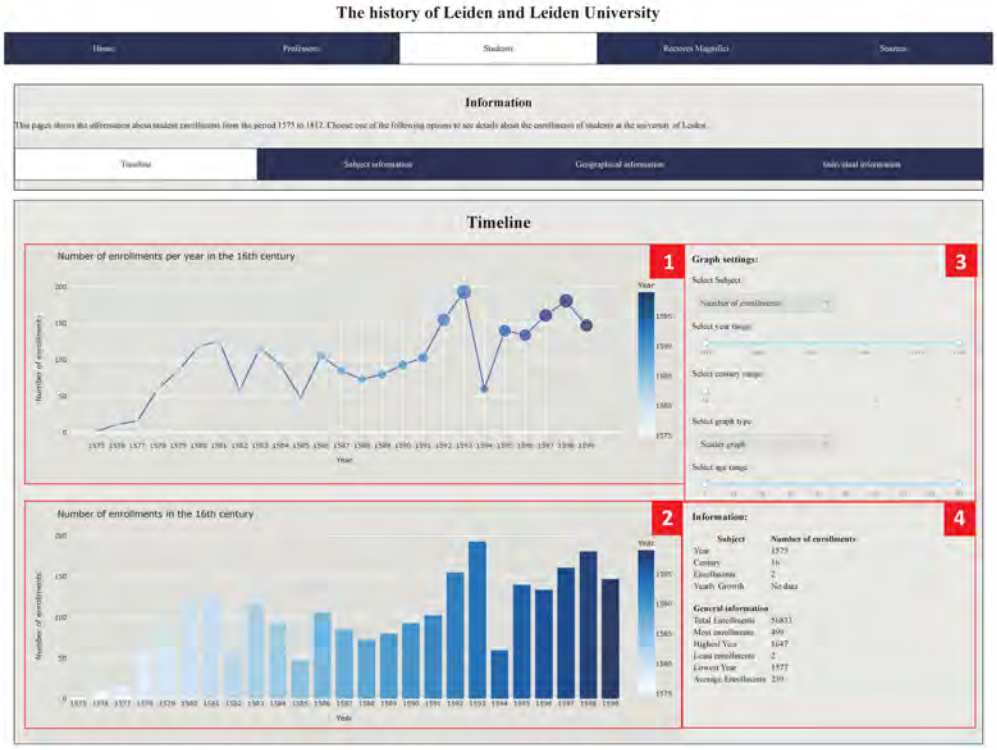



Figure 18: Timeline page of submenu Students

### A.3.1 Year timeline

The uppermost timeline gives an overview of the chosen subject over the selected years. The user can read the amount of enrollments per year for every attribute in the subject. Figure 19 shows the yearly enrollments between the years 1575 and 1599 (the 16th century. The chosen subject is 'Number of Enrollments' which is numerical data. The legend on the right side of the timeline consists of a number scale. An example of categorical data will be shown with another visualisation. The graph can be set to three different types: bar-, line- and scatter graph. All data in the graphs is colour coded to distinguish the unique values in the graph. E.g. when all cities are graphed, every city is a different colour.

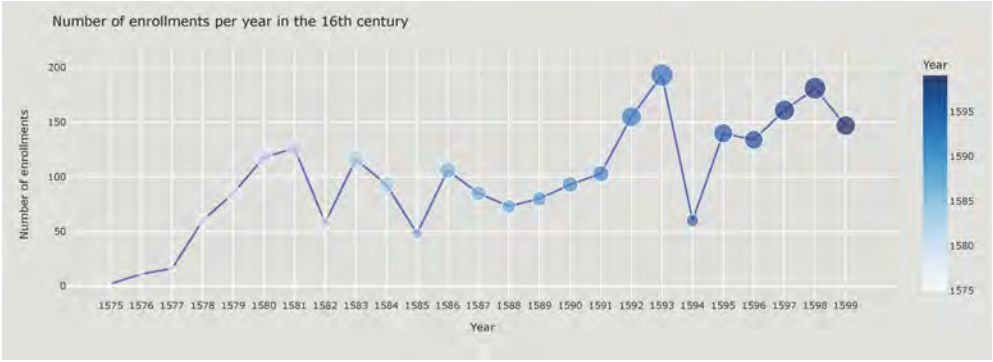



Figure 19: Year timeline for number of enrollments per year

### A.3.2 Century timeline

The bottom timeline mirrors the data from the upper timeline. The century timeline differs as its scale is on a century level instead of individual years. The x-axis is not a set of years but all attributes within the chosen subject. The timeline is a bar graph, which can't be changed. Figure 20 shows the number of enrollments in the 16th century. Because the attributes in the subject are years, the x-axis does show years. If, e.g. the subject country was chosen, the x-axis would be filled with individual countries. If multiple centuries were chosen, the enrollments per century would be stacked on top of each other in blocks representing the century.

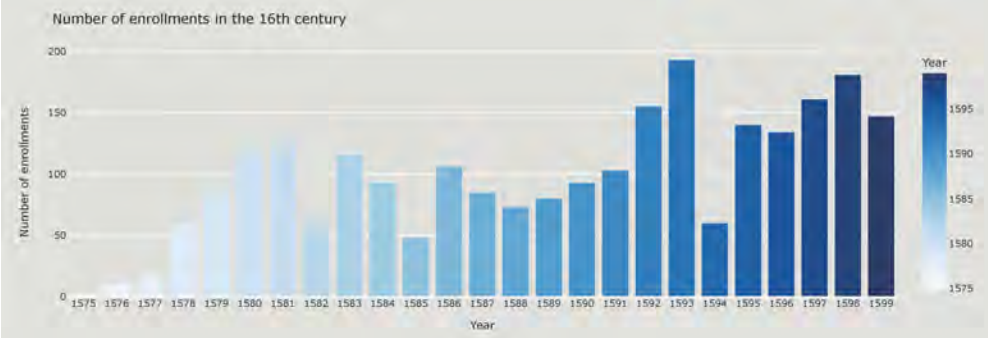



Figure 20: Century timeline for number of enrollments per year

### A.3.3 Timeline settings

The user has the option to change the parameters of the timelines. The following settings are available:

- Select subject: The subject that is represented in the timeline. The user chooses through a drop down bar. Users can type on the drop down bar in order to filter the subjects. Only one subject can be chosen at once.
- Select year range: The range of years that the timeline spans. The available years are dependent on the chosen century. The years are chosen through a range slider. The range slider has two points that determine the minimum and maximum years. Those two points, and the years in between make up the chosen range of years.
- Select century range: A range slider that determines which centuries can be viewed.
- Select graph type: Determines the style of the graph. The options are: Bar graph, Line graph and scatter graph. The type is chosen through a drop down bar. The default option is the line graph.
- Select age range: The age range slider filters the people based on their age at the moment of enrollment/appointment.

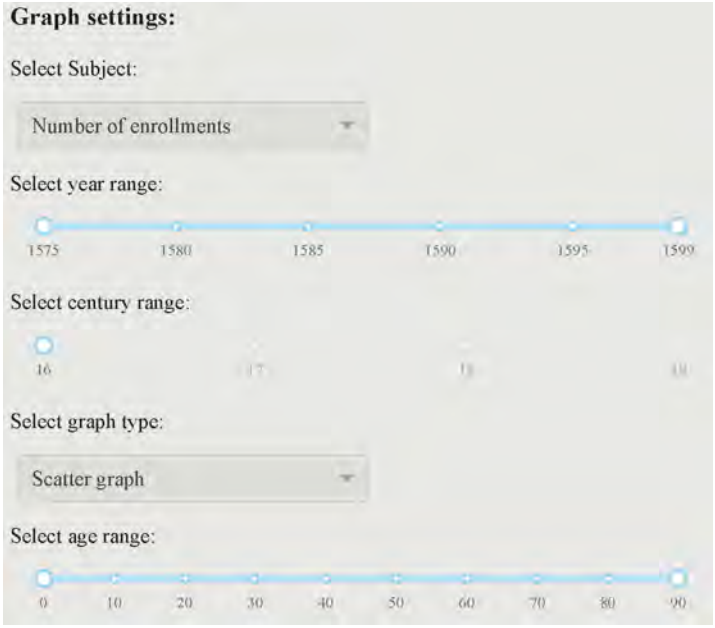



Figure 21: Settings block for the timeline

### A.3.4 Timeline information

The information block has two parts. The first part shows information about the selected point on the timeline. A data point is selected when the user hovers the cursor over the point. The following information is given:

1. The subject that currently is being viewed
2. The attribute that has been selected, this is either a numerical value (e.g. a year or an age) or a categorical value (e.g. a country name, or a city name)
3. The year of the selected point
4. The century of the selected point
5. The numerical difference between the occurrences between the selected point and the point of the previous year

The second part features general information about the chosen subject. This information consists of:

1. The total occurrences in the subject data
2. The highest amount of occurrences of an attribute in a single year
3. The year with the highest amount of occurrences of an attribute
4. The lowest amount of occurrences of an attribute in a single year
5. The year with the lowest amount of occurrences of an attribute

**Information:**

| Subject | Number of enrollments |
|---|---|
| Year | 1575 |
| Century | 16 |
| Enrollments | 2 |
| Yearly Growth | No data |

**General information**

| | |
|---|---|
| Total Enrollments | 56833 |
| Most enrollments | 499 |
| Highest Year | 1647 |
| Least enrollments | 2 |
| Lowest Year | 1577 |
| Average Enrollments | 239 |

Figure 22: Timeline information

## A.4 Subject information

The subject information page gives a deeper insight in the statistics of the subjects compared to the timeline page. Users can sort and filter to create a view of a specific part of the data.
The page has three parts (figure 23)

- Century graph
- Data table
- Information block

### A.4.1 Century graph

The century graph shows the attributes of the subject per century. Every attribute has a different colour. The legend is on the right side of the graph. Figure 24 shows the student enrollments per countries for every century. Categorical data have the added feature that allows for attribute filtering in the legend. Clicking on the attributes within the legend enables/disables them to the graph. Double clicking enables/disables all attributes.

### A.4.2 Data table

The data table has all the information the century graph has in a tabular format. Every column can be filtered or sorted. Every column header has two arrows that sort the table in an ascending or descending order based on that column's data. Filtering can be done by typing in the filter data row directly under the column header row. Filtering might be a bit counter intuitive to some users. The full filtering syntax is found on the Plotly Dash website [Plotly, 2022b]. In general, filtering can be done by writing strings (e.g. netherlands), or writing symbols followed by a string (= 1000). The left side of every filter column has a button that enables/disables case sensitivity. Case sensitivity is turned off by default.
There are two buttons on top of the data table. The first button, named 'Toggle Columns', can

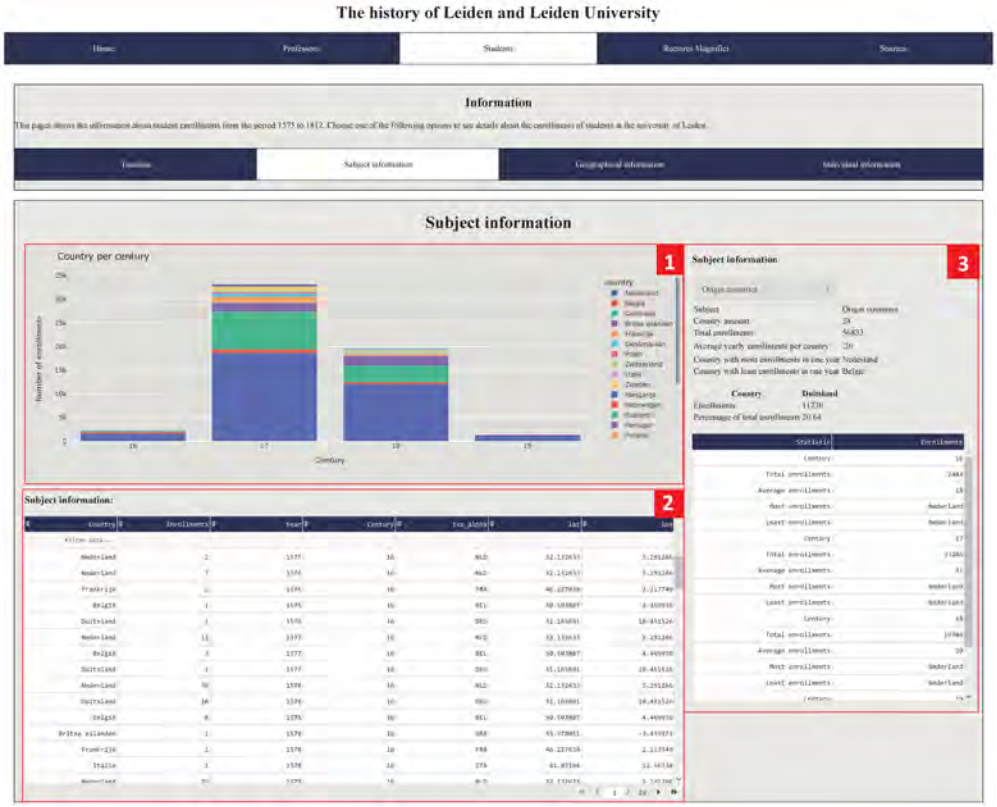



Figure 23: Subject information page of submenu Students

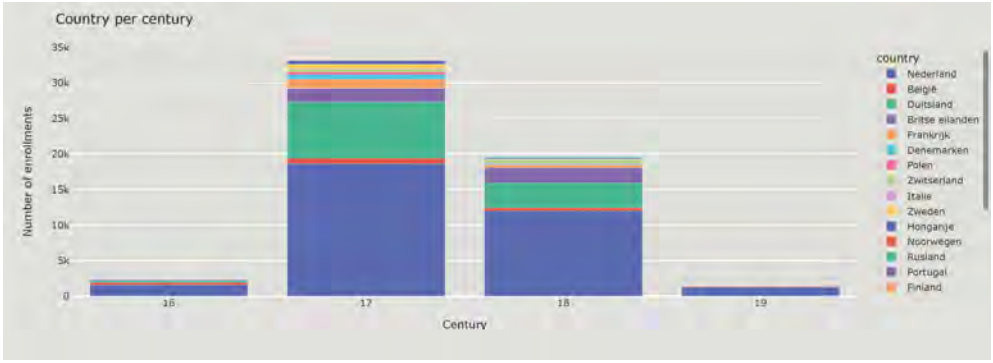



Figure 24: Century graph of student enrollment per countries for every century

be used to toggle columns in the table on or off. The column headers in the table also feature a button that can toggle the column off. The toggle columns button can then be used to show the column again. The second button, named 'Export', is used to export the data table in its current state to a .xlsx format.

### A.4.3 Information block

The third block shows information about the chosen subject. The top of the block houses the drop down bar where users choose which subject is being viewed. More information about the subject is

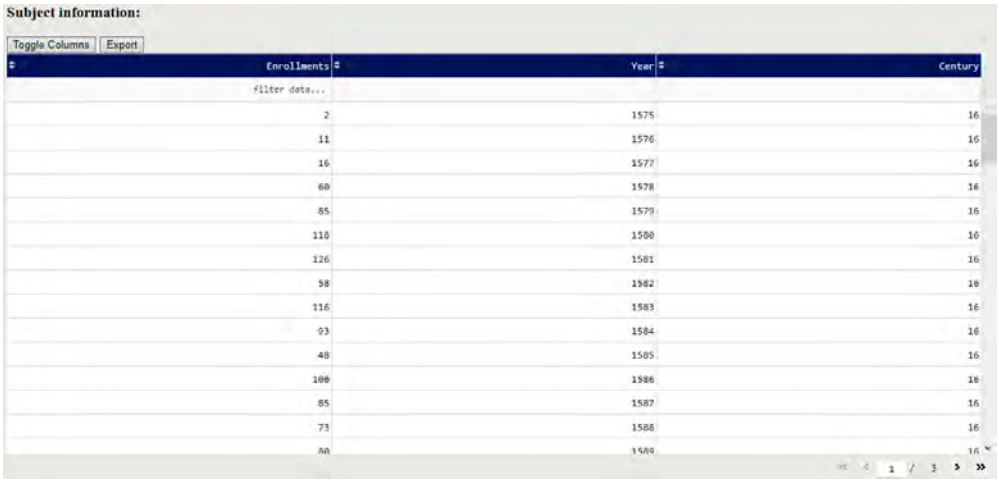



Figure 25: Data table of student enrollment per countries

displayed under the drop down bar:

1. The subject that currently is being viewed
2. The number of attributes the subject has
3. The total amount of occurrences in the subject data
4. The average yearly occurrences per attribute
5. The attribute with the most occurrences in one year
6. The attribute with the least occurrences in one year

Hovering over the century graph shows attribute specific information:

1. The name of the selected attribute
2. The amount of total occurrences of the attribute
3. The percentage of total occurrences that is represented by the attribute

A small data table makes up the bottom part of the block. The table features information per century:

1. The total occurrences
2. The average number of occurrences per attribute
3. The attribute with the most occurrences
4. The attribute with the least occurrences


Figure 26: Subject information of student enrollment per countries

## A.5 Geographical information

The geographical information page shows the geographical representation of the attributes per subject. Three different parts make up the page:

- Geographical map
- Map settings
- Data table

### A.5.1 Geographical map

The geographical map shows the occurrences per subject on a global scale. The choropleth map type is used as its colour coding gives a clear picture of the number of occurrences per country. The legend on the right side shows the numerical equivalent of the colours. The map in figure 28 shows the total amount of student enrollments per country. Some of the maps feature a time function where the map shows the amount of occurrences per year instead of a total number. Pressing the start button starts an automatic year by year animation.

### A.5.2 Map settings

Under the map are the map settings that the users can alter.

- Choose map: There are 5 different types of maps that can be viewed.
    1. Heat map: A Plotly choropleth map

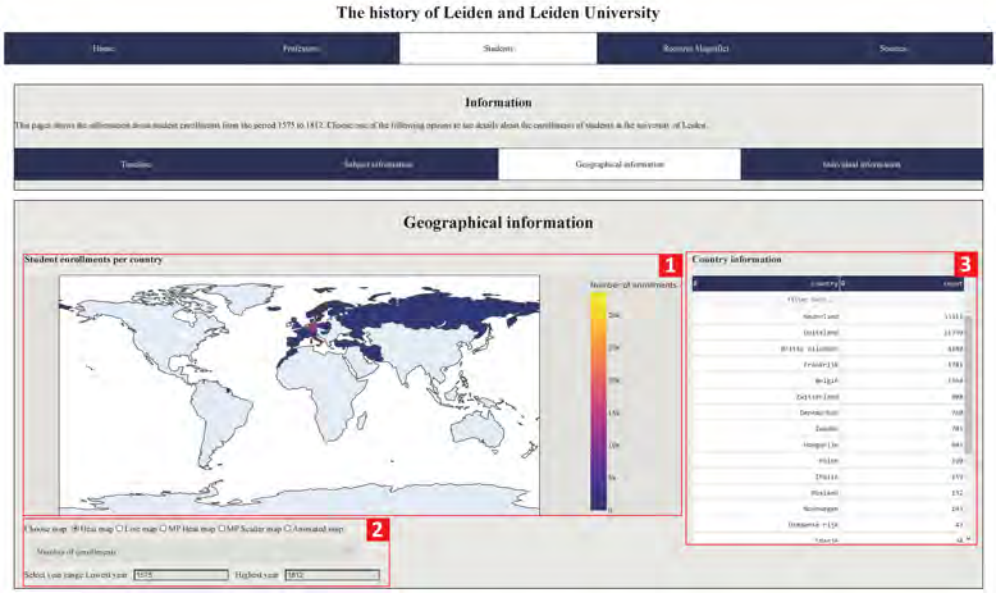



Figure 27: Geographical information page of submenu Students

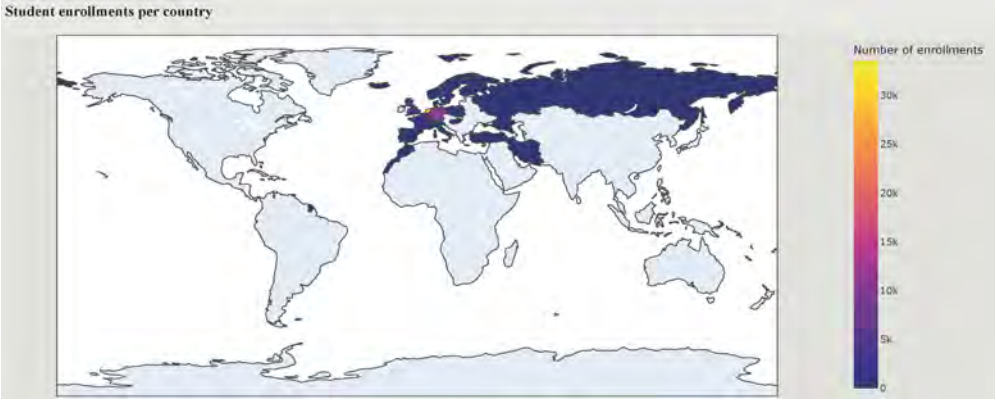



Figure 28: Geographical map

2. Line map: A Plotly map with scatter points that connect to every point to the city of Leiden
3. Mapbox heat map: An animated choropleth map made in Mapbox for a more detailed map
4. Mapbox scatter map: An animated scatterpoint map made in Mapbox
5. Animated map: An animated Plotly choropleth map

- Select subject: Drop down bar that chooses the subject that is viewed
- Select year range: Two input fields denote the range that the map shows. Users have the option to input the lowest and highest years.

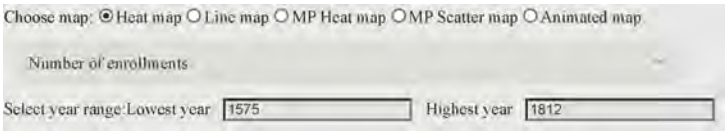



Figure 29: Geographical map

### A.5.3 Data table

The data table on the right side of the page shows the map information in tabular form. The occurrences per country are shown for every year that is selected in the year range. The table features sorting and filtering options.


Figure 30: Geographical map

## A.6 Individual information

The fourth page is the individual information page. Users can find detailed information on the persons featured in the data through manual searching. Figure 31 shows the Student information page. The page has 3 distinct sections:

- Search settings
- Data table
- Selected earch results

### A.6.1 Search settings

The top block of the page shows all parameters the user can fill in to specify their search of a person. Pressing the search button will start a query with the specified parameters. The name input field searches for a corresponding name in both first- and surname data fields. There are two name search functions. 'Contains' means that the inputted string should be somewhere in the full name of the results. E.g. typing in 'John' should give a person named 'Johnson' as result amongst

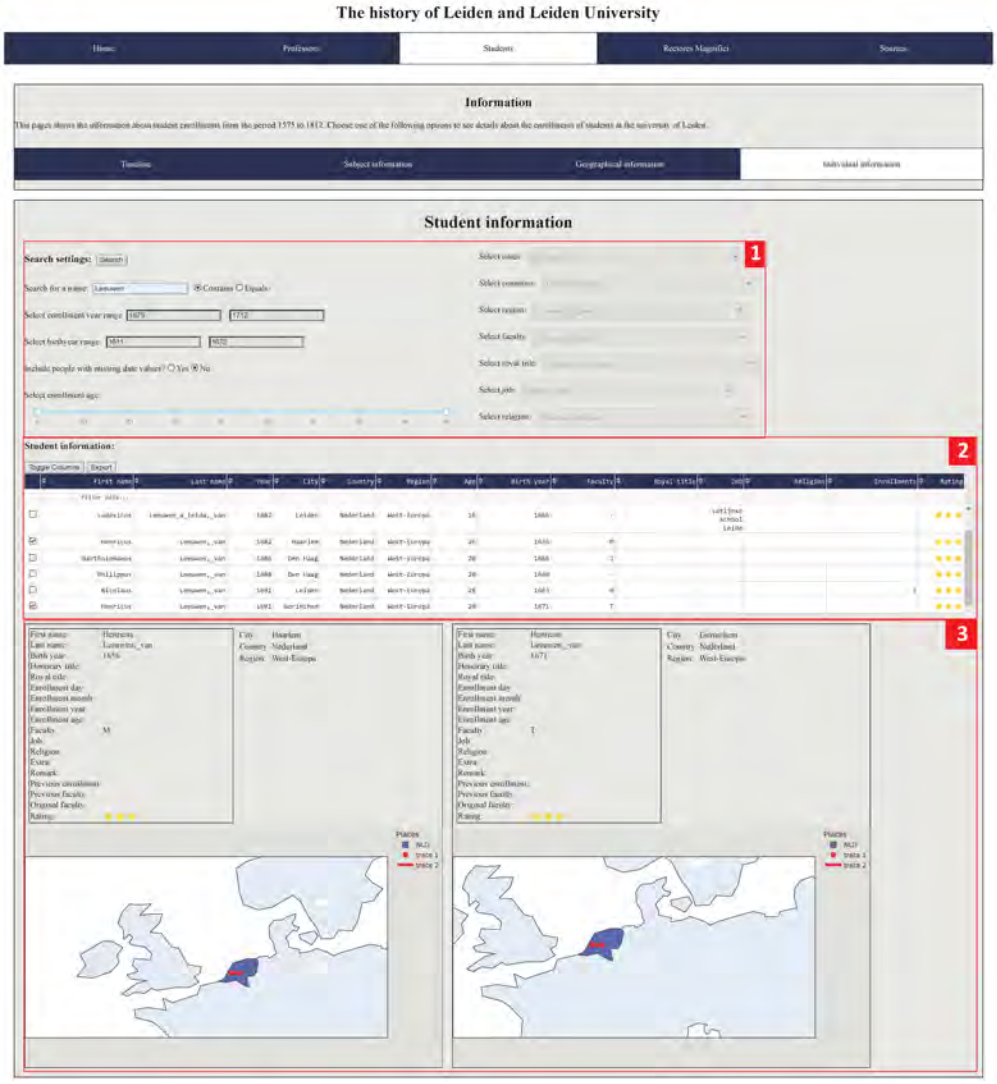



Figure 31: Individual information page of submenu Students

others. 'Equals' means that the inputted string has to be the exact first- or surname of the entries in the search results. There is no case sensitivity in the name search. The settings include a button lets user choose whether they want to include people that have no data in the search fields that were specified. This could be useful if a user is looking for a specific person who's age is not known in the database. The drop down bars allow for multiple options to be chosen. E.g. multiple cities or religions can be chosen to find a broader group of different people.


Figure 32: Search settings in the student menu

### A.6.2 Data table

The data table under the search settings shows the corresponding results after clicking the search button. The table has the same functionalities as the table in the subject information page discussed in section A.4.2. An added functionality is the option to select result entries. The leftmost column includes check boxes. Checking the box of a row will bring up an information box of the person on that row. The boxes will appear in the section under the data table. Deselecting a check box will make the information box disappear.

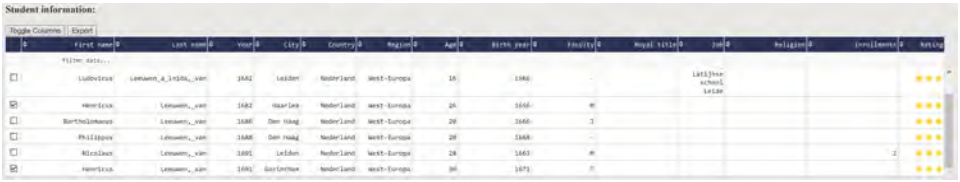



Figure 33: Data table containing search results

### A.6.3 Selected search results

The section under the data table is used for showcasing the selected persons. The box includes all information present in the data table with added geographical and genealogical information in the form of a geographical map that shows the birth country and city; as well as a family tree. The family tree is not yet implemented in the most current version of the dashboard as per date of this thesis. For completeness, two early builds of family tree visualisations have been added to the appendix in figures 55 an 56.

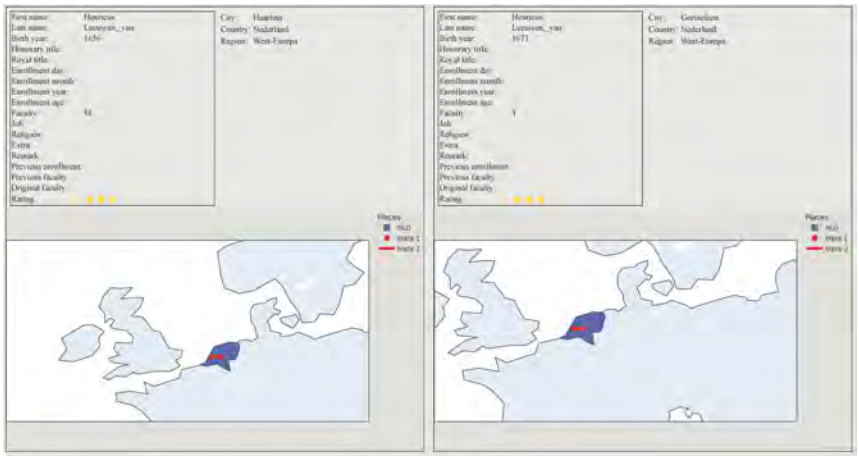


Figure 34: Two selected search results

# B Tables

## B.1 Tables referenced in section 3.1

| General information | | | | |
|---|---|---|---|---|
| Characteristic | Mobility | Segregation | Integration | Personal |
| Nobel price | | | | x |
| Last name | | | | x |
| First name | | | | x |
| Preferred name | | | | Merged with First name |
| Gender | | x | x | x |
| Title | | | x | x |
| Date of birth | x | x | | x |
| Place of birth | x | x | x | x |
| Country of birth | x | x | x | x |
| Date of death | x | x | | x |
| Place of death | x | x | x | x |
| Country of death | x | x | x | x |
| Promotion type | x | | x | x |
| Institution of promotion | x | | x | x |
| Date of promotion | x | | x | x |
| Promotional dissertation | | | | x |

Table 6: Relevance of general information characteristics from professor data set

| Job term information | | | | |
|---|---|---|---|---|
| Characteristic | Mobility | Segregation | Integration | Personal |
| Profession title | | x | x | x |
| Subject area(s) | | x | x | x |
| Date of appointment | x | x | x | x |
| Teaching subject(s) | | x | x | x |
| Faculty | | x | x | x |
| End of employment date | x | x | x | x |
| Reason end of employment | | x | x | x |
| Details | | x | x | x |

Table 7: Relevance of job term information characteristics from professor data set

## B.2 Tables referenced in section 3.2

| General information | | | | |
|---|---|---|---|---|
| Characteristic | Mobility | Segregation | Integration | Personal |
| First name | | | | x |
| Last name | | | | x |
| Place of origin | x | x | x | x |
| Country of origin | x | x | x | x |
| Continent of origin | x | x | x | x |
| Birth year | x | | | x |
| Complement | | x | x | x |
| Remark | | x | x | x |
| Title | | x | x | x |
| Royal status | | x | x | x |
| Job | | x | x | x |
| Religion | | x | x | x |

Table 8: Relevance of general information characteristics from student data set

| Enrollment information | | | | |
|---|---|---|---|---|
| Characteristic | Mobility | Segregation | Integration | Personal |
| Enrollment date | x | x | x | x |
| Enrollment age | x | x | x | x |
| Faculty | | x | x | x |
| Times enrolled | | x | x | x |
| Year of previous enrollment | x | x | x | x |
| Faculty of previous enrollment | | x | x | x |

Table 9: Relevance of enrollment information characteristics from student data set

## B.3 Tables referenced in section 3.4

| Characteristic | Missing rows | Percentage of total rows |
|---|---|---|
| Professor data | | |
| Place of birth | 21 | 0.02 |
| Country of birth | 14 | 0.01 |
| Place of death | 163 | 0.12 |
| Promotion type | 289 | 0.24 |
| Institution of promotion | 291 | 0.25 |
| Date of promotion | 290 | 0.25 |
| Promotional dissertation | 339 | 0.29 |
| Profession title | 7 | 0.01 |
| Subject area | 186 | 0.16 |
| Datum of appointment | 16 | 0.01 |
| Teaching subject | 134 | 0.11 |
| End of employment date | 435 | 0.37 |
| Reason end of employment | 523 | 0.44 |
| Details | 529 | 0.45 |
| Student data | | |
| Place of origin | 19,283 | 0.31 |
| Birth year | 4,903 | 0.08 |
| Complement | 55,653 | 0.91 |
| Remark | 60,559 | 0.99 |
| Job | 56,771 | 0.93 |
| Religion | 61,121 | 100 |
| Enrollment age | 3,616 | 0.06 |

Table 10: Summary of missing data in professor and student data set, rounded to 2 decimals

## B.4 Tables referenced in section 7.2

| Attribute | Attribute type | Data set |
|---|---|---|
| Surname | String | All |
| First name | String | All |
| Nickname | String | Professor |
| Title | String | Professor and Student |
| Nobel Price | String | Professor |
| Death Country | String | Professor and Rector |
| VIAF | String | Professor |
| Handle | String | Professor |
| NTA/PPN | String | Professor |
| AS record number | String | Student |
| Picture link | image file | Rector |
| Picture | URL | Rector |
| Birth Century | String | All |
| Person Gender | String | All |
| Country Origin | String | All |
| Royal Title | String | All |

Table 11: Person object attributes in the NodeGoat LUCD model

| Classifications | Number of categories | Data set |
|---|---|---|
| Century | 4 | All |
| Gender | 2 | All |
| Honours | 11 | Student |
| Origin | 30 | All |
| Professor appointment | 12 | Professor and Rector |
| Religion | 2 | Student |
| Royal title | 5 | Professor and student |
| Student Job | 5 | Student |
| Subject area | 74 | Professor |
| Type of person | 3 | All |

Table 13: Classifications in the NodeGoat LUCD model

| Sub-Object | Time-reference | Geo-reference | Attributes | Data set |
|---|---|---|---|---|
| Birth | Date point | City | | All |
| Death | Date point | City | | All |
| Place of residence | Date range | City | | Professor and Student |
| Promotion Institution | Date point | City | Promotion type, Thesis | Professor |
| Teaching period | Date range | City | Job appointment, Subject area, Acceptance date, Teaching subject, Oration, Oration date, Faculty, Reason end of employment, Details | Professor |
| CV | | | CV description | Professor |
| Student status | Date point | City | Subject name, Age during enrollment, Faculty, Extra, Detail, Gratis, Status, Job, Religion, Previous enrollments, Previous enrollments year, Previous enrollment faculty, First enrollment faculty | Student |
| Rector Magnificus Term | Date range | City | Number of Terms | Rector |

Table 12: Person sub-objects in the NodeGoat LUCD model

## B.5 Tables referenced in section 8

| Feature criteria | | | |
|---|---|---|---|
| Criteria | NodeGoat | Plotly Dash | Score |
| Number of visualisations | 3 (possibility of expansion) | 30+ (possibility of expansion and allows own component creation) | Plotly Dash |
| Visualisation object limit | 100.000 (sub)objects | No limit | Plotly Dash |
| Model creation steps | Requires object modelling, data importing and setting visualisation settings | Requires data importing, data handling, visualisation creation, web page creation | NodeGoat |
| Model customisation | None | All elements of the dashboard can be customised | Plotly Dash |
| Visualisation customisation | Change scope, display settings and minor lay-out options | Full customisation of visualisation | Plotly Dash |
| Database | Automatically initiated with object modelling | Requires external database source | NodeGoat |
| Difficulty | No programming, UI elements for data modelling | All elements have to be manually built, knowledge required of used programming languages | NodeGoat |

Table 14: Comparison of features between the NodeGoat and Plotly Dash prototypes

| Performance criteria | | | |
|---|---|---|---|
| Criteria | NodeGoat | Plotly Dash | Score |
| Model initialisation speed (when working on model) | No initialisation is need as NodeGoat is ran remotely | 55 seconds to start the dashboard app (time depends on machine the app is ran on, irrelevant when deployed on server) | NodeGoat |
| Data importing speed | Importing student data (60.000 rows) takes multiple hours (time could depend on internet speed) | Importing student data (60.000 rows) takes 22 seconds (time depends on machine that runs the importing) | Plotly Dash |
| Data editing | Objects can be edited separately or in bulk with a limit of 128 MB of memory | Objects can be individually edited or in bulk through dataframes and written functions | Plotly Dash |
| Data editing speed | Individual editing: 1 second per object. Bulk editing: 0.005 seconds per object (200 objects per second) | Individual editing: 1 second per object. Bulk editing: 0.0013 seconds per object (769 objects per second) | Plotly Dash |

Table 15: Comparison of features between the NodeGoat and Plotly Dash prototypes

| User experience criteria | | | |
|---|---|---|---|
| Criteria | NodeGoat | Plotly Dash | Score |
| Visualisation loading speed | dependant on visualisation: Geographic: 30 seconds, Network: 108 seconds, Timeline: 6 seconds | Average graph loading: 3 seconds. Maximum graph loading: 8 seconds | Plotly Dash |
| Interactivity | Change view of categories, timeline, selecting objects | Graph parameters, search parameters, hover information | Plotly Dash |

Table 16: Comparison of features between the NodeGoat and Plotly Dash prototypes

# C Figures

All extra figures referred to in this thesis are listed in this section.

## C.1 Figures referenced in section 2.3

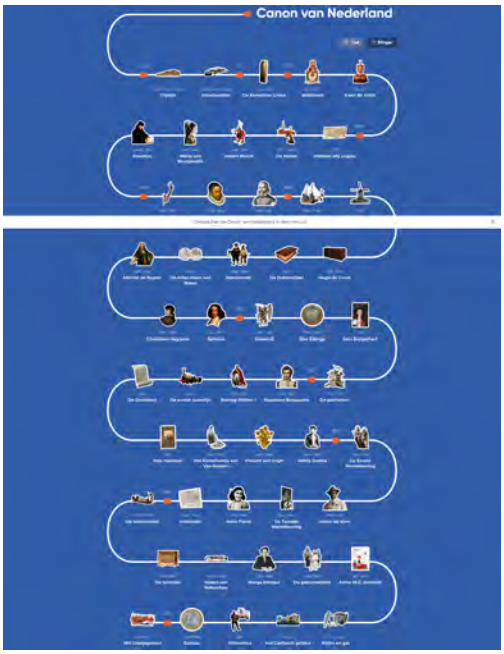



Figure 35: Dutch canon. Source: Canon van Nederland, *https://www.canonvannederland.nl*

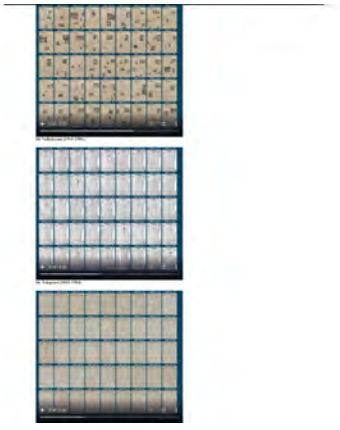


Figure 36: Timelapses of historic newspapers. Source: KB Lab, *https://lab.kb.nl/tool/timelapse − 0*

Figure 37: Example of the Narralyzer tool of the book Blauwe Maandagen. Source: KB Lab *https* : *//lab.kb.nl/tool/narralyzer*


Figure 38: Example of SIAMESE tool pictures of cars in newspapers. Source: KB Lab, *https* : *//lab.kb.nl/tool/siamese*

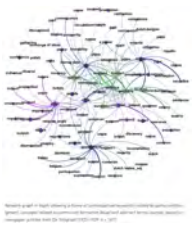


Figure 39: Fame generator example of dutch newspapers from 1925 to 1929. Source: KB Lab, *https* : *//lab.kb.nl/tool/frame − generator*

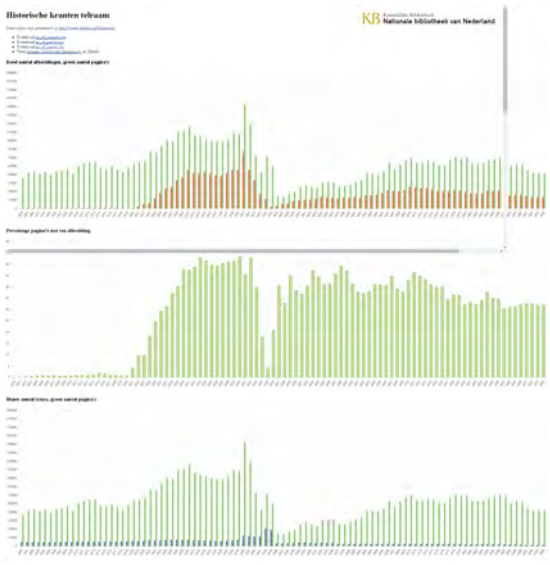



Figure 40: Example of KB Newspapers image count of dutch newspapers. Source: KB Lab, *https* : *//lab.kb.nl/tool/kb − newspapers − image − count*


Figure 41: Example of word search in dutch newspaper. Source: Delpher, *https* : *//www.delpher.nl/over − delpher/wat − is − delpher/delpher − voor − iedereen*


Figure 42: Example of name search in the WieWasWie database. Source: WieWasWie, *https* : *//www.wiewaswie.nl*


Figure 43: Database of Leiden professors from 1575. Source: University of Leiden, *https* : *//hoogleraren.universiteitleiden.nl/s/hoogleraren/search*

## C.2 Figures referenced in section 6.2

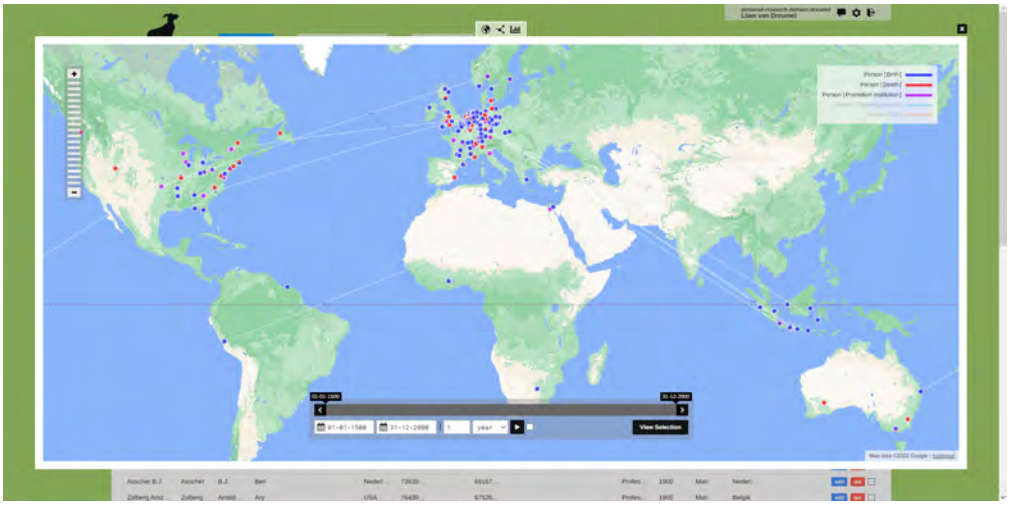


Figure 44: Geographical visualisation of professors from 1500-2000

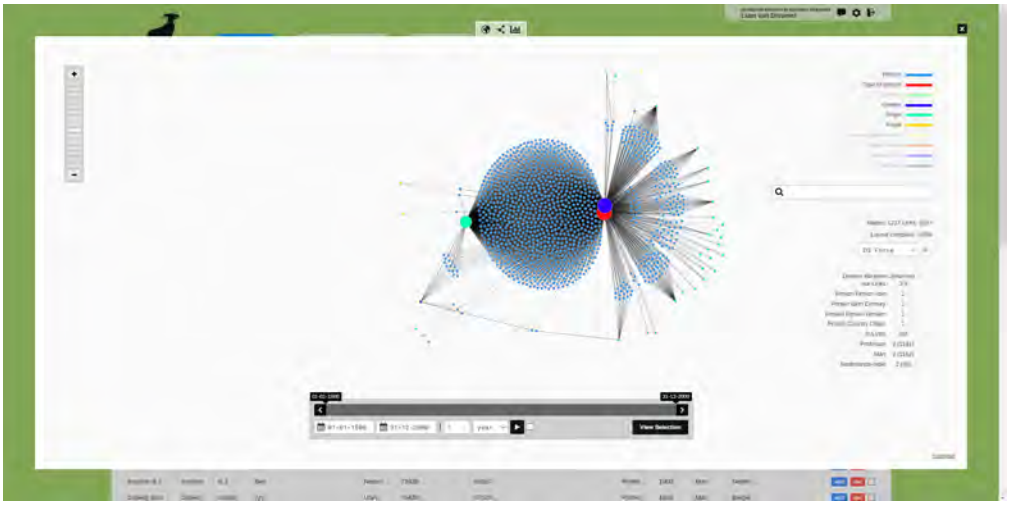


Figure 45: Social visualisation of professors from 1500-2000

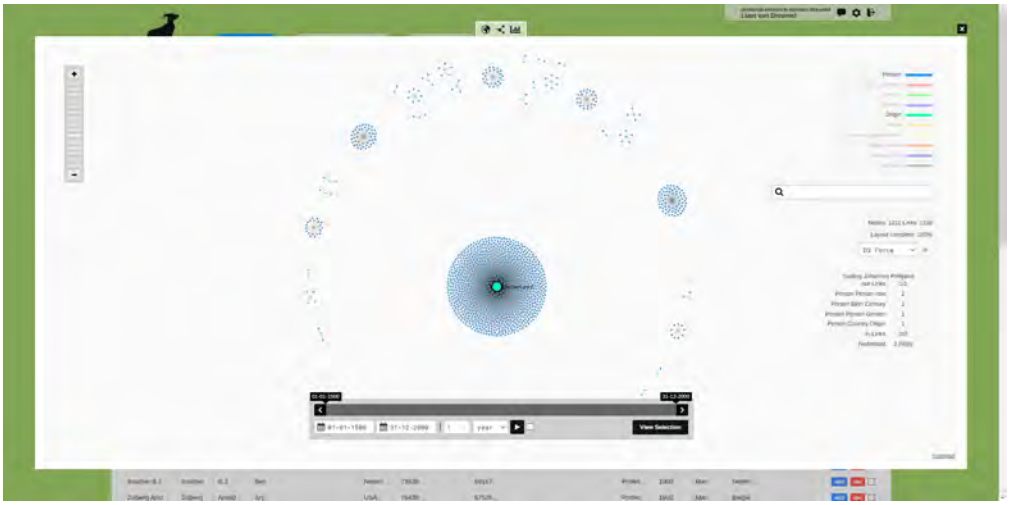


Figure 46: Social visualisation of origins of professors from 1500-2000

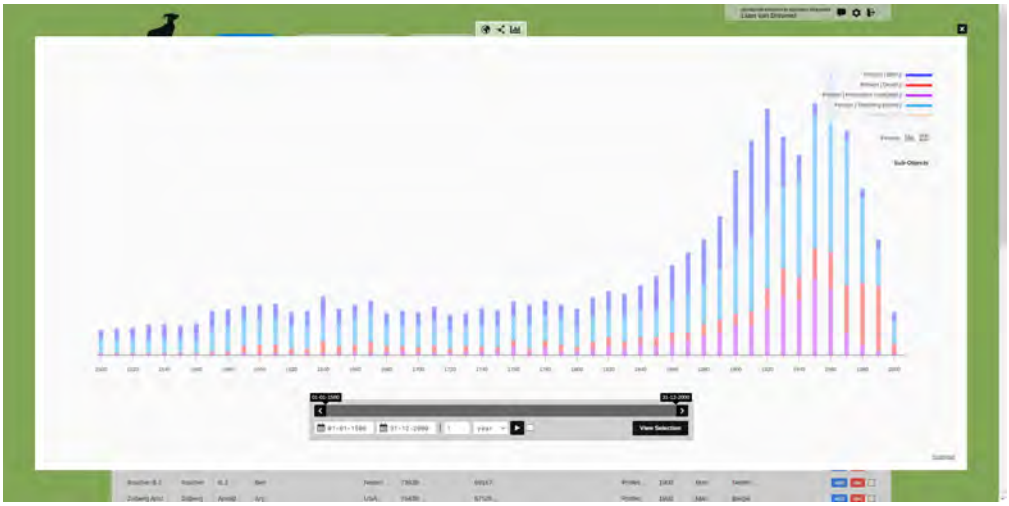


Figure 47: Social visualisation of origins of professors from 1500-2000

## C.3 Figures referenced in section 6.3

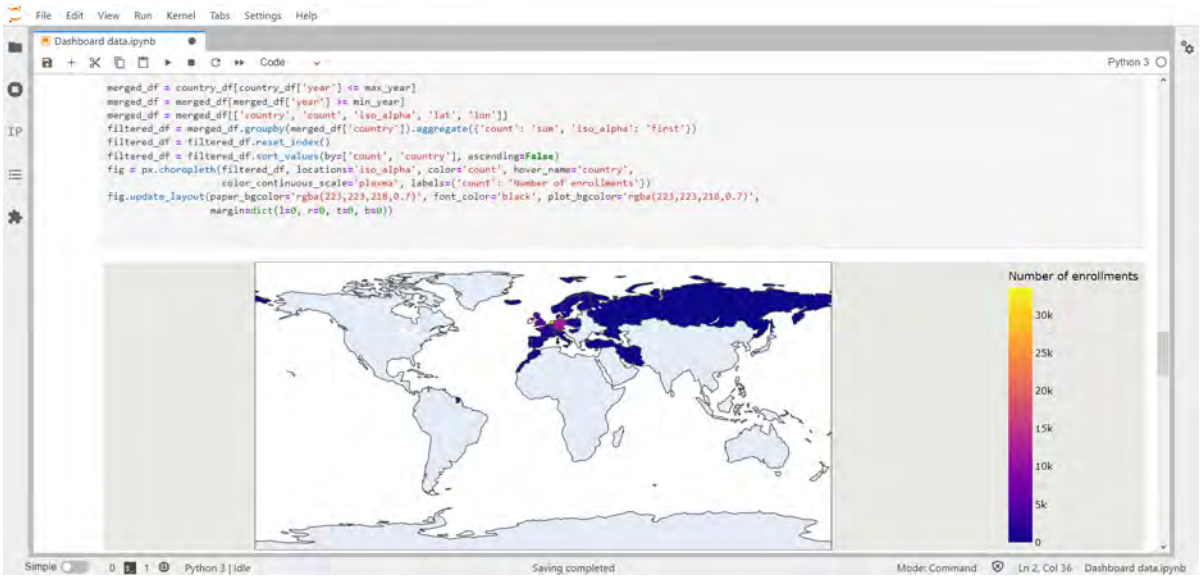
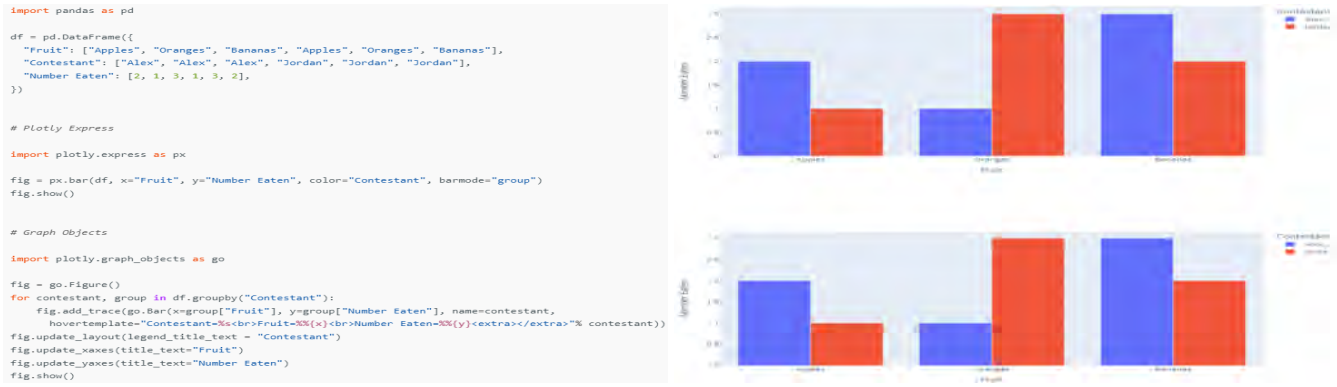



Figure 48: Code of a plotly choropleth map in a JupyterLab notebook

## C.4 Figures referenced in section 7.1



```
import pandas as pd

df = pd.DataFrame({
  "Fruit": ["Apples", "Oranges", "Bananas", "Apples", "Oranges", "Bananas"],
  "Contestant": ["Alex", "Alex", "Alex", "Jordan", "Jordan", "Jordan"],
  "Number Eaten": [2, 1, 3, 1, 3, 2],
})


# Plotly Express

import plotly.express as px

fig = px.bar(df, x="Fruit", y="Number Eaten", color="Contestant", barmode="group")
fig.show()


# Graph Objects

import plotly.graph_objects as go

fig = go.Figure()
for contestant, group in df.groupby("Contestant"):
    fig.add_trace(go.Bar(x=group["Fruit"], y=group["Number Eaten"], name=contestant,
      hovertemplate="Contestant=%s<br>Fruit=%%{x}<br>Number Eaten=%%{y}<extra></extra>"% contestant))
fig.update_layout(legend_title_text = "Contestant")
fig.update_xaxes(title_text="Fruit")
fig.update_yaxes(title_text="Number Eaten")
fig.show()
```

Figure 49: Plotly Express and Graph Objects code comparison of a graph

## C.5 Figures referenced in section 7.2

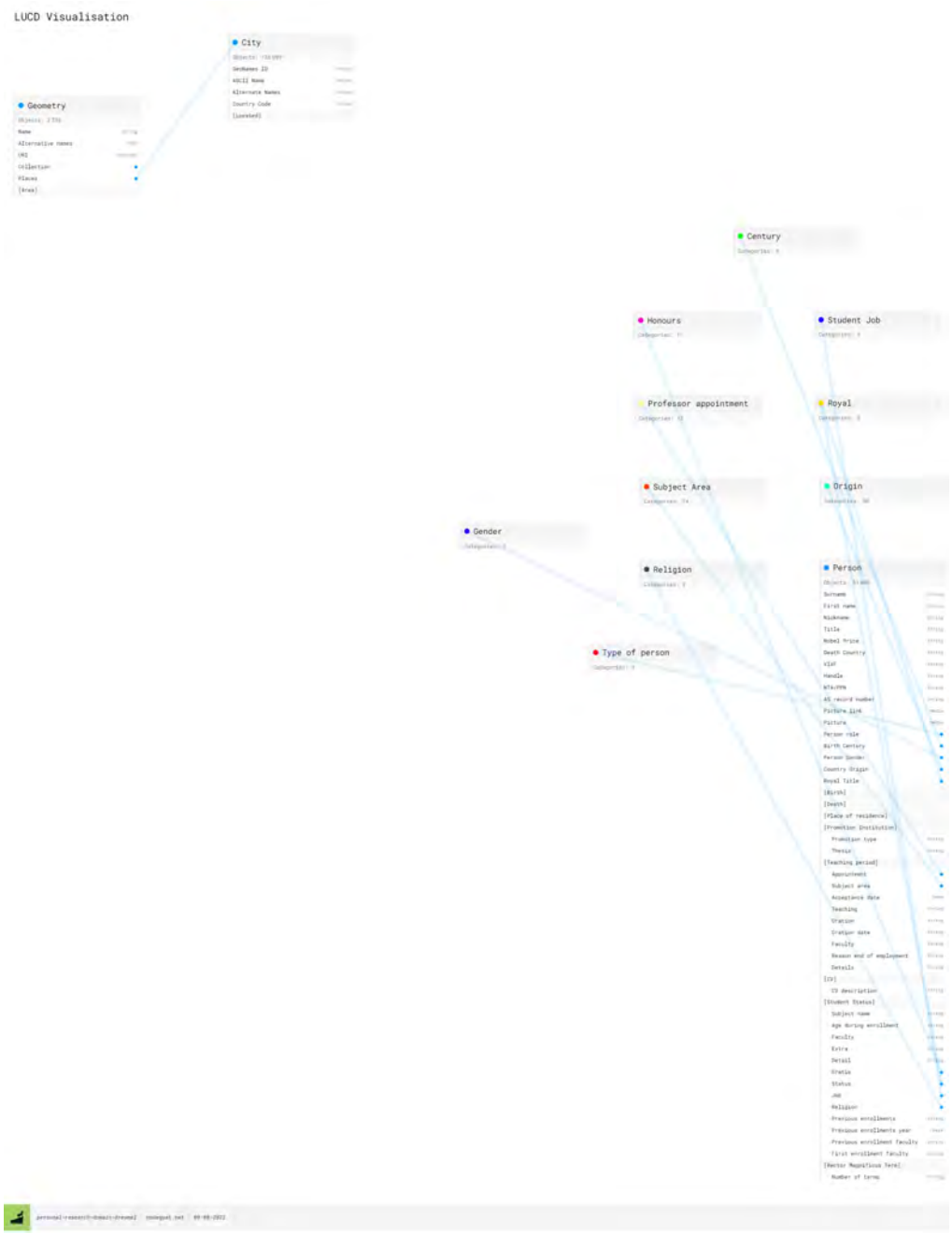



Figure 50: The NodeGoat LUCD model

## C.6 Figures referenced in section 7.3

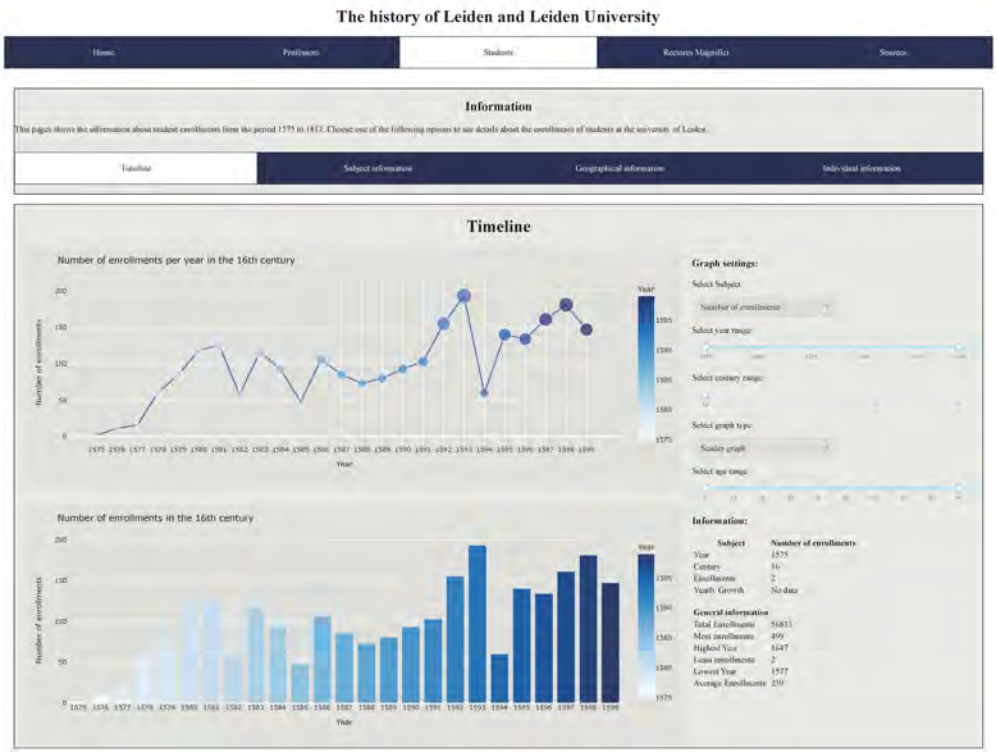



Figure 51: First visualisation page of submenu Students

| | region | count | year | century |
|---|---|---|---|---|
| **0** | West-Europa | 2 | 1575 | 16 |
| **1** | West-Europa | 11 | 1576 | 16 |
| **2** | West-Europa | 16 | 1577 | 16 |
| **3** | West-Europa | 59 | 1578 | 16 |
| **4** | Zuid-Europa | 1 | 1578 | 16 |
| **...** | ... | ... | ... | ... |
| **761** | West-Europa | 140 | 1809 | 19 |
| **762** | West-Europa | 129 | 1810 | 19 |
| **763** | Zuid-Europa | 1 | 1810 | 19 |
| **764** | West-Europa | 111 | 1811 | 19 |
| **765** | West-Europa | 50 | 1812 | 19 |

766 rows × 4 columns

Figure 52: Subject 'region' dataframe, enrollments per year

| | region | count | century |
|---|---|---|---|
| 0 | West-Europa | 2394 | 16 |
| 1 | Noord-Europa | 31 | 16 |
| 2 | Oost-Europa | 16 | 16 |
| 3 | Zuid-Europa | 3 | 16 |
| 4 | West-Europa | 30923 | 17 |
| 5 | Noord-Europa | 1337 | 17 |
| 6 | Oost-Europa | 897 | 17 |
| 7 | Zuid-Europa | 89 | 17 |
| 8 | Overig | 12 | 17 |
| 9 | Afrika | 8 | 17 |
| 10 | West-Europa | 18945 | 18 |
| 11 | Oost-Europa | 383 | 18 |
| 12 | Noord-Europa | 210 | 18 |
| 13 | Zuid-Europa | 132 | 18 |
| 14 | Overig | 35 | 18 |
| 15 | Afrika | 3 | 18 |
| 16 | West-Europa | 1412 | 19 |
| 17 | Zuid-Europa | 2 | 19 |
| 18 | Oost-Europa | 1 | 19 |

Figure 53: Subject 'region' dataframe, enrollments per century

| | region | count |
|---|---|---|
| 0 | West-Europa | 53674 |
| 1 | Noord-Europa | 1578 |
| 2 | Oost-Europa | 1297 |
| 3 | Zuid-Europa | 226 |
| 4 | Overig | 47 |
| 5 | Afrika | 11 |

Figure 54: Subject 'region' dataframe, enrollments per region

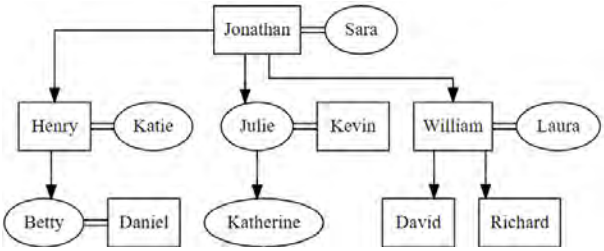



Figure 55: Example of a family tree structure with the graphviz library

Figure 56: Example of a family tree structure with the igraph library