Table of Contents

[1. General Description of the Problem 3](#_Toc438064975)

[2. Project Requirements 4](#_Toc438064976)

[2.1. Which answers am I trying to mine or solve? 4](#_Toc438064979)

[2.2. Formulate a hypothesis question. . 4](#_Toc438064980)

[2.3. What kind of data is being mined? 4](#_Toc438064981)

[2.4. Where does the data originate? 4](#_Toc438064982)

[2.5. Can the data be imported into a database (i.e. for formulating updated queries)? 5](#_Toc438064983)

[2.6. List class and/or concepts related to characteristics and discriminations. 5](#_Toc438064984)

[2.7. Is your project identifying patterns, associations, and/or correlations? 5](#_Toc438064985)

[2.8. Does your dataset contain outliers? If so, please identify them. 5](#_Toc438064986)

[2.9. Was the collected data consider of good or poor quality? 5](#_Toc438064987)

[2.10. Does the data contain missing values or noisy data? 5](#_Toc438064988)

[2.11. Does the dataset contain redundant data? 6](#_Toc438064989)

[2.12. Did you determine the size of your data set? 6](#_Toc438064990)

[3. Data Mining Process 7](#_Toc438064991)

[3.1. Data cleaning 7](#_Toc438064994)

[3.2. Data integration 7](#_Toc438064995)

[3.3. Data selection 7](#_Toc438064996)

[3.4. Data transformation 8](#_Toc438064997)

[3.5. Data mining 9](#_Toc438064998)

[3.6. Pattern evaluation 10](#_Toc438064999)

[3.7. Knowledge presentation 10](#_Toc438065000)

[4. Technology Used 11](#_Toc438065001)

[4.1. Java Technologies 11](#_Toc438065003)

[4.1.1. Java SE Development Kit 8 11](#_Toc438065004)

[4.1.2. Java EE 7 11](#_Toc438065005)

[4.1.3. JSF 2.2 11](#_Toc438065006)

[4.2. Maven Dependencies 11](#_Toc438065007)

[4.2.1. Guava 18.0: Google Core Libraries for Java 11](#_Toc438065008)

[4.2.2. PrimeFaces 5.3 11](#_Toc438065009)

[4.2.3. Junit 4.12 11](#_Toc438065010)

[4.3. JavaScript Libraries 11](#_Toc438065011)

[4.3.1. Data-Driven Documents (D3) 11](#_Toc438065012)

[4.3.2. C3.js | D3-based reusable chart library 11](#_Toc438065013)

[4.3.3. Word Cloud Generator 11](#_Toc438065014)

[4.4. Server and Database 11](#_Toc438065015)

[4.4.1. GlassFish 4.1 11](#_Toc438065016)

[4.4.2. Oracle Database 11g Express Edition 12](#_Toc438065017)

[4.5. Desktop Tools 12](#_Toc438065018)

[4.5.1. IntelliJ IDEA 14.1.5 Ultimate Edition Student License 12](#_Toc438065019)

[4.5.2. NetBeans IDE 8.0.2 12](#_Toc438065020)

[4.5.3. Oracle SQL Developer 4.1.0.19 12](#_Toc438065021)

[4.5.4. Notepad++ v6.8.6 12](#_Toc438065022)

[4.5.5. TortoiseGit 1.8.15.0 12](#_Toc438065023)

[4.5.6. Maven 3.3.3 12](#_Toc438065024)

[4.6. Android Tools 12](#_Toc438065025)

[4.6.1. ForkHub for GitHub 12](#_Toc438065026)

[4.6.2. SGit 12](#_Toc438065027)

[4.6.3. Quoda 12](#_Toc438065028)

[5. Repository and Documentation 13](#_Toc438065029)

[6. Results 14](#_Toc438065030)

[5.1. Distribution of Data 14](#_Toc438065032)

[5.2. Knowledge Presentation 16](#_Toc438065033)

[5.2.1. Age at Death per Birth and Death Year 16](#_Toc438065034)

[5.2.2. Gender and Age Associations 17](#_Toc438065035)

[5.3. Other Results 18](#_Toc438065036)

[5.3.1. Births and Deaths per Month 18](#_Toc438065037)

[5.3.2. Name Frequency 19](#_Toc438065038)

[7. References 20](#_Toc438065039)

# General Description of the Problem

CS7720 Data Mining requires a project worth 30% of the final grade (Saunders, Data Mining Syllabus, 2015). In a previous course, CS7700 Advanced Database Systems, I designed a database based off of my family tree, so for this project I decided to mine that database.

The schema is quite simple, consisting of only two “object” tables and five “relational” tables (see *Figure 1‑1*). The two object tables are *Person* and *Place*. The tables that describe the relations between people are *Father* and *Mother*, which are identical in structure, and *Marriage*. Further describing the *Person*table as well as describing relationships amongst the *Person* and *Place* tables are the *Marriage* and *Event* tables, which are nearly identical except that *Marriage* relates to two people and is for a specific event, while *Event* refers only to one person and can be almost any type of event. Finally, the *Region* table describes the relationship between places.



Figure ‑

# Project Requirements

The course homepage lists several project requirements. Most of these requirements will be listed in this section, along with an appropriate response to that requirement.

1.
2.

## Which answers am I trying to mine or solve?

The most basic question I am trying to answer is how long do people live? Beyond that, how long do people live based on various criteria?

## Formulate a hypothesis question. A hypothesis question contains both a null and alternative question.

For this project, I have tested three hypotheses:

|  |  |
| --- | --- |
| null 1 | A person’s gender has no effect on the age lived to. |
| alternative 1 | Women, on average, live longer, while men, on average, live shorter. |
| null 2 | A person’s birth year has no effect on the age at death. |
| alternative 2 | People live longer when born closer to the present. |
| null 3 | A person’s death year has no effect on the age at death. |
| alternative 3 | The closer someone’s death year is to the present, the longer life that person will have had lived. |

## What kind of data is being mined?

The data is about my ancestors and relatives - when they were born and when they died. Data present that is not being mined includes where they were born, where they died, when and where they were buried, and their relationships to one another.

## Where does the data originate?

The data exists in a single file known as a GEDCOM file, which has an extension of GED. A GEDCOM file is a standard text file, but the exact structure of the file is dependent on the program that creates it. My GEDCOM file was created by Family Tree Maker 2005 (12.0.345 SP1).

The data originally came from many years of research involving interviews with relatives, obituaries, family bibles, Facebook, and other various online resources.

## Can the data be imported into a database (i.e. for formulating updated queries)?

Not only can the data be imported into a database, but it also has been. See section 1 for a description of the schema.

## List class and/or concepts related to characteristics and discriminations.

For what I am mining, my data is characterized by gender and event type.

## Is your project identifying patterns, associations, and/or correlations?

Yes, as explained in question 2.2, which involves the hypotheses questions. I identified if birth year or death year correlate to length of life and if gender is associated with length of life.

## Does your dataset contain outliers? If so, please identify them.

For the concepts I am mining, there are no obvious outliers that would be considered noisy data. Noisy data in this context would include people dying before they were born, and thus having a negative life span, or people living unbelievable long, perhaps much longer than 100 years.

Other outliers do occur in my data, however. The book defines an outlier as “a data object that deviates significantly from the rest of the objects, as if it were generated by a different mechanism” (Han, Kamber, & Pei, 2011, p. 544). In my dataset, there are 17 people who did not live to be a year old, and 42 who did not live to be 10 years old. While the “mechanism” was still death for these individuals, the cause of death was probably different that the rest of the dataset - stillbirth, unique childhood diseases, etc.

## Was the collected data consider of good or poor quality?

Since I have spent years collecting this data, as well as most of this semester cleaning it, it is considered to be good quality.

## Does the data contain missing values or noisy data?

There is no obvious noisy data (see question 2.8).

There is plenty of missing values. Out of 1635 individuals in the database, I was only able to calculate the age at death for 600 of them. This is because the value for birth or death year is missing for some individuals. It could also be because records do not exist or records have not been discovered. Another reason for missing values is that death years do not exist for some people simply because they have not died yet.

Further missing values include people. The dataset is only a subset of my family tree, which, in theory, if complete would include the entirety of humanity.

## Does the dataset contain redundant data? If so, how did you remove duplicated values?

These was no redundant data for the hypotheses I was testing. There was duplicate data in the *Place* table, however. Duplicate values were identified by the SQL query:

SELECT COUNT(NAME), NAME

FROM PLACE

GROUP BY NAME

HAVING COUNT(NAME) > 1

This query retrieves places with duplicate names. This did not necessarily mean they were duplicates, as many places are named the same.

## Did you determine the size of your data set?

Yes. The size of the data can be determined by the following SQL queries:

SELECT COUNT(\*) FROM PERSON;

SELECT COUNT(\*) FROM PLACE;

SELECT COUNT(\*) FROM EVENT WHERE TYPE = 'birth';

SELECT COUNT(\*) FROM EVENT WHERE TYPE = 'death';

The results of these queries reveals that there are 1635 people, 253 places, 946 birth records, and 627 death records.

# Data Mining Process

1.

The book breaks down the data mining, or knowledge discovery, process into seven steps (Han, Kamber, & Pei, 2011, pp. 6-8). These steps, as well as how I did the steps, are as follows:

1.

## Data cleaning

This is accomplished “by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies” (Han, Kamber, & Pei, 2011, p. 85). Because of my years of work with this data, there was no need to clean that data which I was mining. However, during this semester I noticed some inconsistencies with places. Each place has a name and zero or one regions. A region is another place, which in turn can have zero or one regions as well. There were multiple times in my dataset where a single place appeared to be multiple places because the data “skipped” a region in one entry, but not another. For instance, the following appeared as two entries, but was actually one:

|  |
| --- |
| Dayton, Ohio |
| Dayton, Montgomery County, Ohio |

Since the data originated in a text file, this was fixed by replacing all occurrences of *Dayton, Ohio* with *Dayton, Montgomery County, Ohio*. Conversely, occurrences of *Dayton, Montgomery County, Ohio* could have been replaced with *Dayton, Ohio*, which would have eliminated the problem of it appearing as two places, however, that would have removed additional information.

## Data integration

This is the process of including data from multiple sources - typically multiple databases or files (Han, Kamber, & Pei, 2011, p. 85). For this project, I only used a single source, which was my GEDCOM file. This file, however, is hand-integrated from multiple sources over the years.

## Data selection

Here only data relevant to the mining tasks are selected (Han, Kamber, & Pei, 2011, p. 8). Since I was only interested in age, death, birth, and gender, I only needed to select from the *Person* and *Event* tables. Furthermore, I only needed to select from those tables where I could calculate the age (i.e., those people who had both a known birth and death). I decided I did not need an exact age (e.g., days, months, and years) and that years only would suffice. Therefore, I ignored the *month* and *day* attributes in the *Event* table.

I selected these values by creating views in my database. A view is essentially a stored SQL *select* statement that can be queried as if it were a table. The query to calculate the ages appears as follows:

SELECT B.PERSON\_ID,

 (D.YEAR - B.YEAR) AS AGE,

 B.YEAR AS BIRTH\_YEAR,

 D.YEAR AS DEATH\_YEAR

FROM EVENT B, EVENT D

WHERE B.PERSON\_ID = D.PERSON\_ID

AND B.TYPE = 'birth'

AND D.TYPE = 'death'

AND B.YEAR IS NOT NULL

AND D.YEAR IS NOT NULL

The retention of *person id*, *birth year*, and *death year* in the query were necessary to help relate *age* to further data mining queries.

## Data transformation

Here “data are transformed or consolidated into forms appropriate for mining” (Han, Kamber, & Pei, 2011, p. 112). GEDCOM files are not optimized for storage in a relational database. I wrote custom JPA entities (which are POJOs - plain old Java objects - with annotations and/or XML files describing the relationship between the entities and the database schema) and a Java importer class to load the data.

An example for an entry for an individual in the GEDCOM file appears as follows:

0 @I0036@ INDI

1 NAME William Zenos /Thoroman/

1 SEX M

1 BIRT

2 DATE 4 MAR 1827

2 PLAC Ohio

1 DEAT

2 DATE JAN 1900

2 PLAC Dayton, Montgomery County, Ohio

The name had to have the forward slash (/) removed before entry into the database. Sex was easy to determine, as the only options were “M” or “F”, meaning male or female, respectively.

Date parsing was slightly more complicated. The string after the “2 DATE” was split into tokens by spaces. Each token was checked if it contained one or two numbers, four numbers, or neither of those. If it was the first option, the day field was filled. If it was the second option, the year field was filled. Otherwise, it was assumed to be the month, which was matched to the corresponding Month enum in Java 8’s date API. The Java code for the token parsing is as follows:

if ( token.matches("[0-9]{1,2}") ) {

 int number = Integer.parseInt(token);

 event.setDay(number);

} else if ( token.matches("[0-9]{4}") ) {

 int number = Integer.parseInt(token);

 event.setYear(number);

} else {

 for ( Month month : Month.values() ) {

 if ( month.name().startsWith(token) ) {

 event.setMonth(month);

 break;

 }

 }

}

Finally, places had to be parsed. Similar to how dates were parsed, the place string was split into tokens, but this time by the comma and space. Since places are dependent on their regions, the tokens were processed in reverse. The last token was assumed to be a US state unless it matched a predefined list of countries.

## Data mining

The actual process where statistics and other methods are used to discover patterns. In my data set I mined the view described in section 3.3. I did this by creating two new views that referenced that view (called *Age View*). They are called *Age to Birth Year View* and *Age to Death Year View* and are defined by the two following select statements, respectively:

SELECT AVG(AGE) AVG\_AGE, MEDIAN(AGE) MEDIAN\_AGE, BIRTH\_YEAR

FROM AGE\_VIEW

GROUP BY BIRTH\_YEAR

ORDER BY BIRTH\_YEAR

SELECT AVG(AGE) AVG\_AGE, MEDIAN(AGE) MEDIAN\_AGE, DEATH\_YEAR

FROM AGE\_VIEW

GROUP BY DEATH\_YEAR

ORDER BY DEATH\_YEAR

I also did simple queries to find associations with gender:

SELECT AVG(AGE) FROM AGE\_VIEW, PERSON

WHERE AGE\_VIEW.PERSON\_ID = PERSON.ID

GROUP BY GENDER

HAVING GENDER = {MALE / FEMALE}

SELECT AVG(AGE) FROM AGE\_VIEW

## Pattern evaluation

In this step it is determined whether patterns are interesting or not. If a pattern is subjective and depends on criteria specified by the miner.

To determine if age and birth / death year correlate, I graphed the average and median ages per year (which is actually the next step, see section 3.7).

To determine the interestingness of the average age of gender, I first found the absolute difference between average overall age and the specified gender’s average age, then found the percent difference between the those two statistics.

## Knowledge presentation

This final step presents results to the user. The results can be in the form of graphs or other pictorial representations of the data, or just the most interesting raw results.

I present the results of this project in section 5 on page 12.

# Technology Used

1.

## Java Technologies

### Java SE Development Kit 8

<http://docs.oracle.com/javase/8/docs/index.html>

### Java EE 7

<http://docs.oracle.com/javaee/7/index.html>

### JSF 2.2

<https://javaserverfaces.java.net/docs/2.2/>

## Maven Dependencies

### Guava 18.0: Google Core Libraries for Java

<https://github.com/google/guava>

### PrimeFaces 5.3

<http://www.primefaces.org/>

### Junit 4.12

<http://junit.org/>

## JavaScript Libraries

### Data-Driven Documents (D3)

<http://d3js.org/>

### C3.js | D3-based reusable chart library

<http://c3js.org/>

### Word Cloud Generator

<https://www.jasondavies.com/wordcloud>

## Server and Database

### GlassFish 4.1

<https://glassfish.java.net/>

### Oracle Database 11g Express Edition

<http://www.oracle.com/technetwork/database/database-technologies/express-edition/overview/index.html>

## Desktop Tools

### IntelliJ IDEA 14.1.5 Ultimate Edition Student License

<https://www.jetbrains.com/idea>

### NetBeans IDE 8.0.2

<https://netbeans.org/>

### Oracle SQL Developer 4.1.0.19

<http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index-097090.html>

### Notepad++ v6.8.6

<https://notepad-plus-plus.org/>

### TortoiseGit 1.8.15.0

<https://tortoisegit.org/>

### Maven 3.3.3

<https://maven.apache.org/>

## Android Tools

### ForkHub for GitHub

<https://play.google.com/store/apps/details?id=jp.forkhub>

### SGit

<https://play.google.com/store/apps/details?id=me.sheimi.sgit>

### Quoda

<http://www.getquoda.com/>

# Repository and Documentation

A GitHub repository, which includes a README on technical aspects of the program, can be found at:

<https://github.com/hendrixjoseph/FamilyTree>

Standard Maven documentation, as well as JavaDoc and JSF tag documentation, can be found at:

<http://hendrixjoseph.github.io/FamilyTree/>

# Results

Many of these results can be viewed interactively at:

<http://hendrixjoseph.github.io/tags/family_tree>

1.

## Distribution of Data

Before we look at interestingness measures, it is important to understand how the data is distributed so that we have an idea of the quality of data. Since the primary attribute being studied is age, first we will look at that distribution. *Figure 5‑1* is a bar graph showing the distribution of ages while *Table 5‑1* show the five-number summary plus average. Of the 600 ages calculated, half (300) were older than 69 years and 25% (155) were 80 years or older. Ages existed from as short as 0 years to as longs as 98 years.



Figure ‑

|  |  |
| --- | --- |
| Measure | Value (years) |
| Minimum | 0 |
| Q1 | 48 |
| Median | 69.5 |
| Q2 | 80 |
| Maximum | 98 |
| Average | 61.27 |

Table ‑

The second measure is births and deaths related to year. Therefore it would be interesting to see the distribution of births and deaths per year. Displaying all the years in a single chart proved unfeasible, as the years ranged from 1745 to 2014 (and thus 270 years). Therefore, the years were binned into decades as displayed in *Figure 5‑2*. Here we see that most births and deaths occurred from about the 1860s to the 1920s.



Figure ‑

The final attribute we are considering is gender. According to *Figure 5‑3*, the dataset consists of 48.9% females and 51.1% males, which is very close to a 1:1 ratio.



Figure ‑

## Knowledge Presentation

### Age at Death per Birth and Death Year

First I looked at average and median age per birth year. As *Figure 5‑4* shows, there was no correlation. At this point I decided it would be trivial to investigate average and median age per death year. *Figure 5‑5* shows a general increase in the average and median as time goes on.



Figure ‑



Figure ‑

### Gender and Age Associations

On page 291 of the book (Han, Kamber, & Pei, 2011), the following example was presented as an example of an association rule that exhibited exceptional behavior:

$$sex=female=>meanwage=\$7.90/hr(overall\\_mean\\_wage=\$9.02/hr)$$

Equation ‑

To see if exception behavior existed for age based on age in my dataset, I calculated the mean age for both males and females. As indicated by *Equation 5‑1* I also calculated the overall mean age. I compared the mean age for a given gender to the overall mean age using three methods. *Equation 5‑2* is the absolute difference, or just difference. *Equation 5‑3* is the percent difference. Finally, *Equation 5‑4* is just simply the percent.

$$mean age-overall mean age$$

Equation ‑

$$\frac{mean age-overall mean age}{overall mean age}$$

Equation ‑

$$\frac{mean age}{overall mean age}$$

Equation ‑

Finally, the results of the equations are presented in *Table 5‑2*. We see that females live a litter more than half a year longer than average, while males live a litter more than half a year less than average.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Gender | Average age | Difference | Percent difference | Percent |
| male | 60.73 years | 0.55 years | 0.89% | 99.11% |
| female | 61.92 years | 0.65 years | 1.06% | 101.06% |
| overall | 61.27 years | 0.0 years | 0.0% | 100.0% |

Table ‑

## Other Results

In addition to the results presented above, three other charts were generated during this project.

### Births and Deaths per Month

*Figure 5‑6* is similar to *Figure 5‑2* on page 14, except that it displays number of births and deaths per month. However it does not contain the same subset of individuals. Instead it contains the subset of individuals whose month of birth *or* death is known.



Figure ‑

### Name Frequency

For *Figure 5‑7* and *Figure 5‑8* the top 20 first and last names, respectively, are presented as word clouds. The font size is equal to the number of occurrences of that name. The color and rotation are random, however.



Figure ‑



Figure ‑

# References

Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Waltham, Massachusetts: Elsevier.

Saunders, E. (2015, Fall). Data Mining Syllabus. Dayton, OH: Department of Computer Science and Engineering, Wright State University.