

Between-census population interpolation for Amsterdam, 1850-1929: An R tutorial

Katalin Buzasi
LUMC and Radboud University Nijmegen
email: k.buzasi@let.ru.nl
website: katalinbuzasi.weebly.com
github: <https://github.com/KatiBuzasi>

April 2021

Contents

1	Introduction	2
2	Linear and exponential interpolation	2
2.1	STEP 1: create and set the working directory and import necessary libraries in the R environment	2
2.2	STEP 2: load in/import and inspect the data	3
2.3	STEP 3: building and executing the interpolation process	4
2.4	STEP 4: saving tables	7
3	Aggregating age groups	8
4	Comparison with data reported in the yearbooks	10

1 Introduction

The aim of this tutorial is to reconstruct the annual age-specific population by gender for Amsterdam between 1850 and 1929 to be used in the research project titled "Lifting the burden of disease. The modernisation of health in the Netherlands: Amsterdam 1854-1940"¹. The results of this exercise can be used as population at risk in our various subprojects or for calculating age-specific mortality rates by gender. This tutorial might be useful for students and researchers in general history and social and economic history who often need to rely on simple techniques to obtain age-specific population between censuses which are usually available only once in every ten years.

This paper relies on two main data sources. First, the population by age (single years between 0 and 100) by gender for 1849, 1859, 1869, 1879, 1889, 1899, 1909, 1920 and 1930 is available for Amsterdam from the censuses². Secondly, the "Statistische Jaarboeken" (statistical yearbooks) provide population counts by gender (but not for age groups) for each year starting in 1849³. The underlying data in xlsx format and the codes in R script format are available on the author's github page.

In this paper I use the "*interp*" command of the R library "*DemoTools*" by Riffe et al. (2019) to obtain the desired population numbers. This package is suitable to conduct simple linear and exponential interpolation. (Actually, the package offers a third method called *power*, but we do not use it in this tutorial.) An alternative population estimate for Amsterdam by age group (but not gender) for the same period is available in van Leeuwen and Oeppe (1993).

This paper is structured as follows. Section 2 shows in detail how to perform the interpolation using the "*DemoTools*" library. Section 3 demonstrates how to aggregate our interpolated population data into larger age groups. Section 4 compares the interpolated data with population numbers in the yearbooks.

Note: this tutorial does not contain the instructions of how to install R and R studio and how to get started with these softwares. That information can be found at <https://www.r-project.org/> and <https://rstudio.com/>.

2 Linear and exponential interpolation

2.1 STEP 1: create and set the working directory and import necessary libraries in the R environment

You can create a folder, the so called working directory⁴, on your own computer where you store the datasets and eventually you export the interpolated datasets and potential

¹<https://www.ru.nl/rich/our-research/research-groups/radboud-group-historical-demography-family-history/current-research-projects/lifting-burden-disease/>

²<http://www.volkstellingen.nl/nl/index.html>

³The yearbooks are available in pdf format on the website of the Amsterdam City Archive at <https://data.amsterdam.nl/publicaties/zoek/>. The yearbook 1905-09 contains a large summary table on page 33.

⁴Working directory is the folder which is directly connected to your coding environment. If you set this at the beginning of your work/code/script, you can load in/import and export/save the data and graphs without typing in the whole path.

graphs. For instance, I create a new empty subfolder ("*interpolation*") in the already existing "*C:/Documents/project*" folder and I copy my underlying datasets ("*men_census_pop.xlsx*", "*women_census_pop.xlsx*" and "*JB_pop.xlsx*") there in *xlsx* or any other preferred format (e.g. *csv*).

After setting the working directory, we load in the necessary *R* libraries. To be able to load in the necessary libraries, you have to install them first. This installation needs to be done only once. While the *readxl*, *writexl*, *ggplot2*, *reshape2* and *dplyr* libraries simply can be installed using the code *install.packages()*, the *DemoTools* library should be installed manually following the instructions at <https://timriffe.github.io/DemoTools/>.

```
1
2 # set working directory
3 ## giving the path to the folder you want to work in
4 path="c:/Documents/project/interpolation"
5 setwd(path)
6
7 # loading in necessary libraries
8 ## for being able to import xlsx files
9 #install.packages("readxl")
10 library(readxl)
11
12 ## for being able to save our tables in xlsx format
13 #install.packages("writexl")
14 library(writexl)
15
16 ## package for interpolation
17 library(DemoTools)
18
19 ## package for visualization
20 #install.packages("ggplot2")
21 library(ggplot2)
22
23 ## package for reshaping tables from wide to long
24 #install.packages("reshape2")
25 library(reshape2)
26
27 ## package required for making age group aggregations
28 #install.packages("dplyr")
29 library(dplyr)
```

2.2 STEP 2: load in/import and inspect the data

Then, we import the two basic datasets ("*men_census_pop.xlsx*" and "*women_census_pop.xlsx*") from the working directory where they are stored into the *R*-studio environment. These datasets are imported as data frames and they contain the population by age and gender from the census years. These two datasets exclude those whose age is unknown. Rows refer to age groups and columns refer to census years (this is the layout required by the *interp* package).

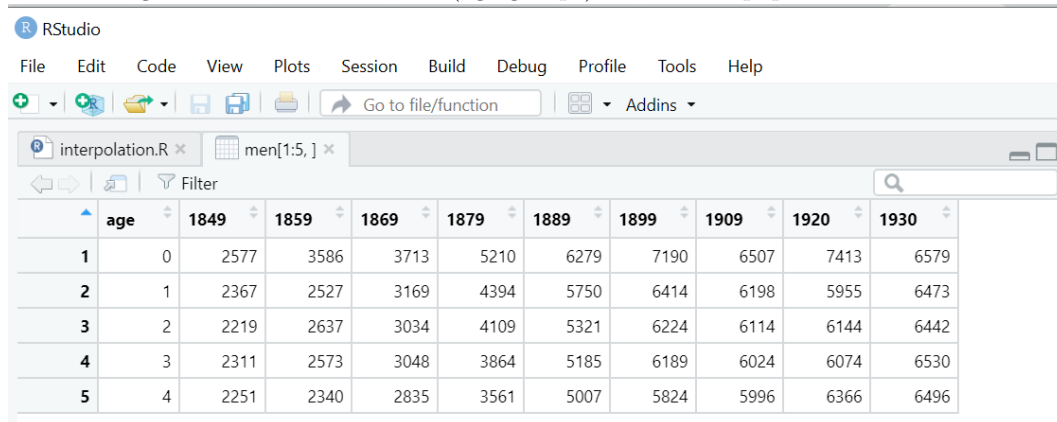
```
1 # importing the datasets
2 men <- read_xlsx("men_census_pop.xlsx")
3 women <- read_xlsx("women_census_pop.xlsx")
```

You can inspect the dataset itself or the first five lines, for instance, with the *View()* command.

```
1 # inspecting the first five rows
2 View(men[1:5,])
3 View(women[1:5,])
```

The imported dataset should look like as shown in Figure 1.

Figure 1: The first five rows (age groups) of the male population dataset

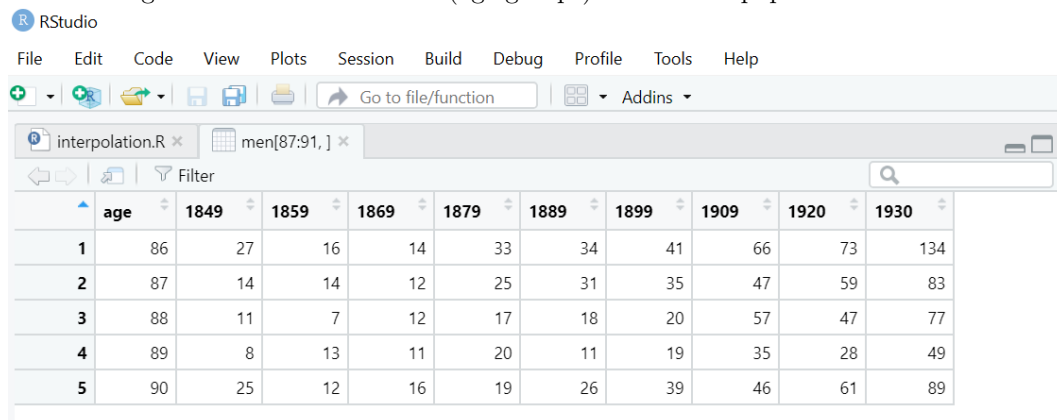


	age	1849	1859	1869	1879	1889	1899	1909	1920	1930
1	0	2577	3586	3713	5210	6279	7190	6507	7413	6579
2	1	2367	2527	3169	4394	5750	6414	6198	5955	6473
3	2	2219	2637	3034	4109	5321	6224	6114	6144	6442
4	3	2311	2573	3048	3864	5185	6189	6024	6074	6530
5	4	2251	2340	2835	3561	5007	5824	5996	6366	6496

It is important to note here that we have made a small change to the original census data. The original data provide population count by single-year age group between 0 and 100. However, since the count of some age groups above 90 is 0, which cannot be handled by the exponential interpolation method, we combined the age groups between 90 and 100 to avoid these problematic zeros (see Figure 2). Thus, the last row, 90, is actually the age group of 90 and above. The basic datasets, "men_census_pop.xlsx" and "women_census_pop.xlsx", already contain this combined group instead of the single-year age groups between 90 and 100.

```
1 # inspecting the last five rows
2 View(men[87:91,])
3 View(women[87:91,])
```

Figure 2: The last five rows (age groups) of the male population dataset



	age	1849	1859	1869	1879	1889	1899	1909	1920	1930
1	86	27	16	14	33	34	41	66	73	134
2	87	14	14	12	25	31	35	47	59	83
3	88	11	7	12	17	18	20	57	47	77
4	89	8	13	11	20	11	19	35	28	49
5	90	25	12	16	19	26	39	46	61	89

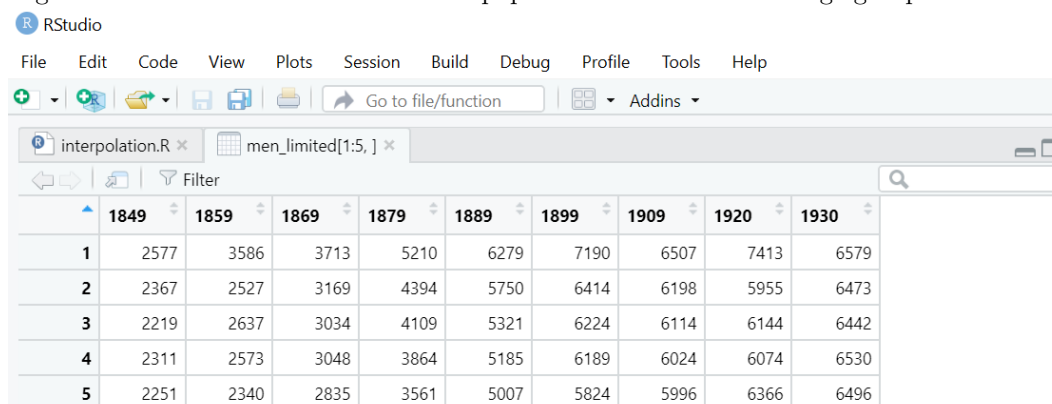
2.3 STEP 3: building and executing the interpolation process

The **syntax** of the *interp* code is *interp(data, datesIn, datesOut, method, rule)*.

data refers to the dataset which contains the census population count by age in columns, exactly the structure of our underlying datasets (see again Figure 1 and 2). However, we need only the part of the datasets that contain the population numbers (see more on this later) only. Thus, we create two new datasets without the age group information. The datasets should look as shown in Figure 3. The actual single-year age groups are saved as a separate vector because we need to use them later.

```
1 # dataframes containing only the population numbers by year
2 men_limited <- men[,2:10]
3 women_limited <- women[,2:10]
4 View(men_limited[1:5,])
5 View(women_limited[1:5,])
6
7 # saving age vector
8 age <- men$age
```

Figure 3: The last five rows of the male population dataset without age group information



	1849	1859	1869	1879	1889	1899	1909	1920	1930
1	2577	3586	3713	5210	6279	7190	6507	7413	6579
2	2367	2527	3169	4394	5750	6414	6198	5955	6473
3	2219	2637	3034	4109	5321	6224	6114	6144	6442
4	2311	2573	3048	3864	5185	6189	6024	6074	6530
5	2251	2340	2835	3561	5007	5824	5996	6366	6496

datesIn refer to the exact dates of the censuses. This can be in "YYYY-MM-DD" format or in decimal format (see at <https://rdr.io/github/timriffe/DemoTools/man/interp.html>). The code can be executed only if the number of the columns in the data equals the number of dates in the *datesIn* vector, that is why we needed to delete the age group column earlier.

```
1 # exact dates of censuses
2 datesIn <- c("1849-11-19", "1859-12-13", "1869-12-01", "1879-12-31",
3 "1889-12-31", "1899-12-31", "1909-12-31", "1920-12-31", "1930-12-31")
```

datesOut refer to the dates of the population interpolations. If you want to use the population data to calculate age-specific death rates, you might be interested in the mid-year population (as of 1st July). If it fits your purposes, you can set the dates at the very end of the year (31st Dec).

```
1 # the dates that we need the population estimates for
2 datesOut <- c("1850-07-01", "1851-07-01", "1852-07-01", "1853-07-01",
3 "1854-07-01", "1855-07-01", "1856-07-01", "1857-07-01", "1858-07-01",
4 "1859-07-01", "1860-07-01", "1861-07-01", "1862-07-01", "1863-07-01",
5 "1864-07-01", "1865-07-01", "1866-07-01", "1867-07-01", "1868-07-01",
6 "1869-07-01", "1870-07-01", "1871-07-01", "1872-07-01", "1873-07-01",
7 "1874-07-01", "1875-07-01", "1876-07-01", "1877-07-01", "1878-07-01",
8 "1879-07-01", "1880-07-01", "1881-07-01", "1882-07-01", "1883-07-01",
```

```

9 "1884-07-01", "1885-07-01", "1886-07-01", "1887-07-01", "1888-07-01",
10 "1889-07-01", "1890-07-01", "1891-07-01", "1892-07-01", "1893-07-01",
11 "1894-07-01", "1895-07-01", "1896-07-01", "1897-07-01", "1898-07-01",
12 "1899-07-01", "1900-07-01", "1901-07-01", "1902-07-01", "1903-07-01",
13 "1904-07-01", "1905-07-01", "1906-07-01", "1907-07-01", "1908-07-01",
14 "1909-07-01", "1910-07-01", "1911-07-01", "1912-07-01", "1913-07-01",
15 "1914-07-01", "1915-07-01", "1916-07-01", "1917-07-01", "1918-07-01",
16 "1919-07-01", "1920-07-01", "1921-07-01", "1922-07-01", "1923-07-01",
17 "1924-07-01", "1925-07-01", "1926-07-01", "1927-07-01", "1928-07-01",
18 "1929-07-01")

```

method refers to the interpolation method, which can be linear, exponential or power.

rule refers to the approximation rule regarding the dates. See more on this at <https://rdr.io/github/timriffe/DemoTools/man/interp.html>.

First, we conduct the linear interpolation on the male dataset.

```
1 men_linear <- interp(men_limited, datesIn, datesOut, "linear", rule = 2)
```

The resulting dataset looks as shown in Figure 4.

Figure 4: The interpolated (linear) male population dataset

	1850.49589041096	1851.49589041096	1852.49726775956	1853.49589041096	1854.49589041096	18
1	2638.5177	2738.7586	2839.1375	2939.2403	3039.4812	
2	2376.7550	2392.6505	2408.5679	2424.4415	2440.3370	
3	2244.4850	2286.0120	2327.5961	2369.0659	2410.5928	
4	2326.9739	2353.0027	2379.0674	2405.0604	2431.0893	
5	2256.4262	2265.2681	2274.1221	2282.9518	2291.7937	
6	2316.6576	2330.7648	2344.8915	2358.9793	2373.0866	
7	2346.7681	2349.6492	2352.5342	2355.4113	2358.2923	

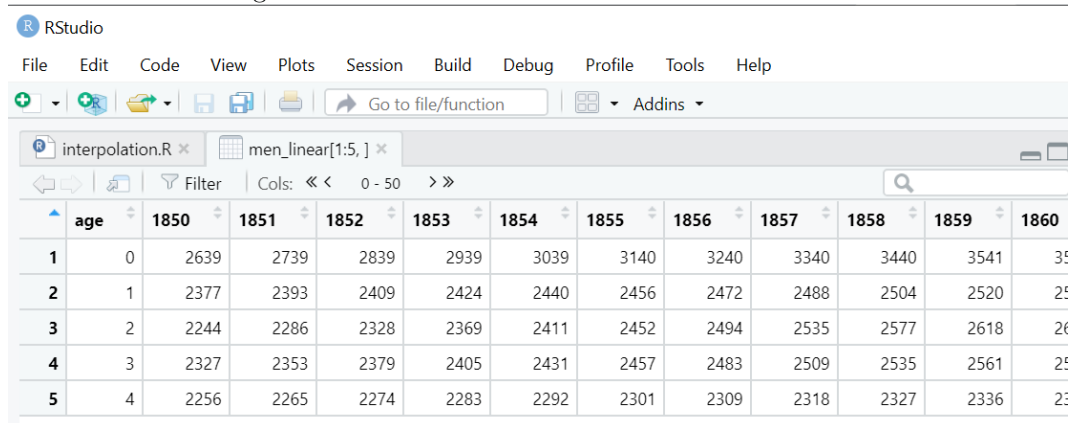
The resulting table looks quite good, however, we would like to apply some changes. First, we round the population numbers to integers. Second, we change the column names, which are currently the dates we set in the datesOut vector in decimal format, to years (knowing that each year refers to the mid-year population). Third, we include the age groups column that was removed earlier. Finally, we convert the table into dataframe. The first five rows of the final dataset are shown in Figure 5.

```

1 men_linear <- round(men_linear, 0)
2 colnames(men_linear) <- seq(from=1850, to=1929, by=1)
3 men_linear <- cbind(age, men_linear)
4 men_linear <- as.data.frame(men_linear)
5
6 # inspecting the first five rows
7 View(men_linear[1:5,])

```

Figure 5: The first five rows of the final male dataset



	age	1850	1851	1852	1853	1854	1855	1856	1857	1858	1859	1860
1	0	2639	2739	2839	2939	3039	3140	3240	3340	3440	3541	3641
2	1	2377	2393	2409	2424	2440	2456	2472	2488	2504	2520	2536
3	2	2244	2286	2328	2369	2411	2452	2494	2535	2577	2618	2659
4	3	2327	2353	2379	2405	2431	2457	2483	2509	2535	2561	2587
5	4	2256	2265	2274	2283	2292	2301	2309	2318	2327	2336	2345

The codes for the female population and the exponential interpolation method is shown below.

```

1 # women linear
2 women_linear <- interp(women_limited, datesIn, datesOut, "linear", rule = 2)
3 women_linear <- round(women_linear, 0)
4 colnames(women_linear) <- seq(from=1850, to=1929, by=1)
5 women_linear <- cbind(age, women_linear)
6 women_linear <- as.data.frame(women_linear)
7
8 # men exponential
9 men_exponential <- interp(men_limited, datesIn, datesOut,
10 "exponential", rule = 2)
11 men_exponential <- round(men_exponential, 0)
12 colnames(men_exponential) <- seq(from=1850, to=1929, by=1)
13 men_exponential <- cbind(age, men_exponential)
14 men_exponential <- as.data.frame(men_exponential)
15
16 # women exponential
17 women_exponential <- interp(women_limited, datesIn, datesOut,
18 "exponential", rule = 2)
19 women_exponential <- round(women_exponential, 0)
20 colnames(women_exponential) <- seq(from=1850, to=1929, by=1)
21 women_exponential <- cbind(age, women_exponential)
22 women_exponential <- as.data.frame(women_exponential)

```

2.4 STEP 4: saving tables

We save our tables in xlsx format in the working directory. You can export the files in other formats, such as csv, if you prefer. See the help menu in R studio.

```

1 # saving our tables in xlsx format
2 write_xlsx(men_linear, "men_midyear_linear.xlsx", col_names = TRUE)
3 write_xlsx(women_linear, "women_midyear_linear.xlsx",
4 col_names = TRUE)
5 write_xlsx(men_exponential,
6 "men_midyear_exponential.xlsx", col_names = TRUE)
7 write_xlsx(women_exponential,
8 "women_midyear_exponential.xlsx", col_names = TRUE)

```

3 Aggregating age groups

The previous tables contain the population interpolation by single-year age group (age 0, age 1, age 2 etc.), except for the last one (discussed earlier). However, we sometimes want age groups of higher aggregation, such as age 0-4, age 5-9, age 10-14, or age groups of unequal length, such as age 0, age 1-4, age 5-14 etc. The codes below show how we can obtain these aggregations using the interpolation tables from the previous subsection. Here we have to note that although our underlying data in the previous subsection had single-year age groups, the *interp* command works with any arbitrary age grouping. This means that if you have census data for combined age groups only, you can do the interpolation the same way as shown in the previous subsection.

The first step is to create a new column, let us call it *age_group*, in the interpolation tables which indicates which aggregated age group the single-age group belongs to. We create the following age groups: 0-4, 5-14, 15-24, 25-34, 35-44, 45-54 and 55 and above. These group labels are created with the *cut* command. After checking that the grouping is indeed what we want (see Figure 6), we delete the original *age* column because we do not need it anymore and the code which perform the summation by age group by year requires a table which does not include unnecessary columns.

```
1 # men_linear
2 ## creating age groups
3 men_linear$age_group <- cut(men_linear$age, c(0, 5, 15, 25, 35, 45, 55, 100), right=FALSE)
4 # inspecting the first 16 rows and the first, second, third and last
5 # column of the resulting table
6 View(men_linear[1:16, c(1,2,3,4,5,82)])
7 # deleting the first column
8 men_linear <- men_linear[, -1]
```

Figure 6: Inspecting age group labels

	age	1850	1851	1852	1853	age_group
1	0	2639	2739	2839	2939	[0,5]
2	1	2377	2393	2409	2424	[0,5]
3	2	2244	2286	2328	2369	[0,5]
4	3	2327	2353	2379	2405	[0,5]
5	4	2256	2265	2274	2283	[0,5]
6	5	2317	2331	2345	2359	[5,15]
7	6	2347	2350	2353	2355	[5,15]
8	7	2368	2374	2380	2385	[5,15]
9	8	2362	2380	2397	2414	[5,15]
10	9	2292	2311	2330	2349	[5,15]
11	10	2244	2257	2269	2281	[5,15]
12	11	2103	2109	2114	2120	[5,15]
13	12	2024	2015	2007	1998	[5,15]
14	13	2021	2034	2047	2060	[5,15]
15	14	1917	1944	1970	1997	[5,15]
16	15	1815	1854	1894	1934	[15,25]

Then, we sum the individual age groups in each aggregated age groups by year. A subset of the resulting table is shown in Figure 7.


```

1 ## age group aggregation
2 aggr_men_linear <- men_linear %>%
3   group_by(age_group) %>%
4   summarise(across(everything(), sum))

```

Figure 7: An excerpt of the *men_linear* table aggregated by age group

	age_group	1850	1851	1852	1853	1854	1855	1856	1857	1858	1859	1860	1861	1862	1863	1
1	[0,5]	11843	12036	12229	12420	12613	12806	12998	13190	13383	13576	13780	13996	14208	14423	
2	[5,15]	21995	22105	22212	22318	22425	22534	22638	22746	22853	22961	23121	23320	23520	23717	
3	[15,25]	18736	18914	19094	19272	19452	19630	19811	19989	20166	20349	20460	20523	20584	20646	
4	[25,35]	17022	17021	17023	17024	17026	17025	17027	17030	17031	17032	17094	17211	17326	17440	
5	[35,45]	13241	13519	13794	14074	14350	14628	14905	15181	15459	15737	15858	15853	15847	15843	
6	[45,55]	9947	10018	10090	10161	10234	10304	10377	10448	10521	10592	10777	11057	11336	11617	
7	[55,100]	9640	9750	9857	9967	10074	10184	10289	10396	10505	10614	10779	10989	11197	11408	

The codes below do the same process for the remaining three interpolation tables obtained in the previous subsection.

```

1 # women linear
2 women_linear$age_group <- cut(women_linear$age,
3   c(0, 5, 15, 25, 35, 45, 55, 100), right=FALSE)
4 women_linear <- women_linear[,-1]
5
6 aggr_women_linear <- women_linear %>%
7   group_by(age_group) %>%
8   summarise(across(everything(), sum))
9
10 # men exponential
11 men_exponential$age_group <- cut(men_exponential$age,
12   c(0, 5, 15, 25, 35, 45, 55, 100), right=FALSE)
13 men_exponential <- men_exponential[,-1]
14
15 aggr_men_exponential <- men_exponential %>%
16   group_by(age_group) %>%
17   summarise(across(everything(), sum))
18
19 # women exponential
20 women_exponential$age_group <- cut(women_exponential$age,
21   c(0, 5, 15, 25, 35, 45, 55, 100), right=FALSE)
22 women_exponential <- women_exponential[,-1]
23
24 aggr_women_exponential <- women_exponential %>%
25   group_by(age_group) %>%
26   summarise(across(everything(), sum))

```

It is possible to save these tables in xlsx format using the *write_xlsx* command again.

```

1 # saving these tables in the working directory in xlsx format
2 write_xlsx(aggr_men_linear, "aggr_men_linear.xlsx", col_names = TRUE)
3 write_xlsx(aggr_women_linear, "aggr_women_linear.xlsx", col_names = TRUE)
4 write_xlsx(aggr_men_exponential, "aggr_men_exponential.xlsx", col_names = TRUE)
5 write_xlsx(aggr_women_exponential, "aggr_women_exponential.xlsx", col_names = TRUE)

```

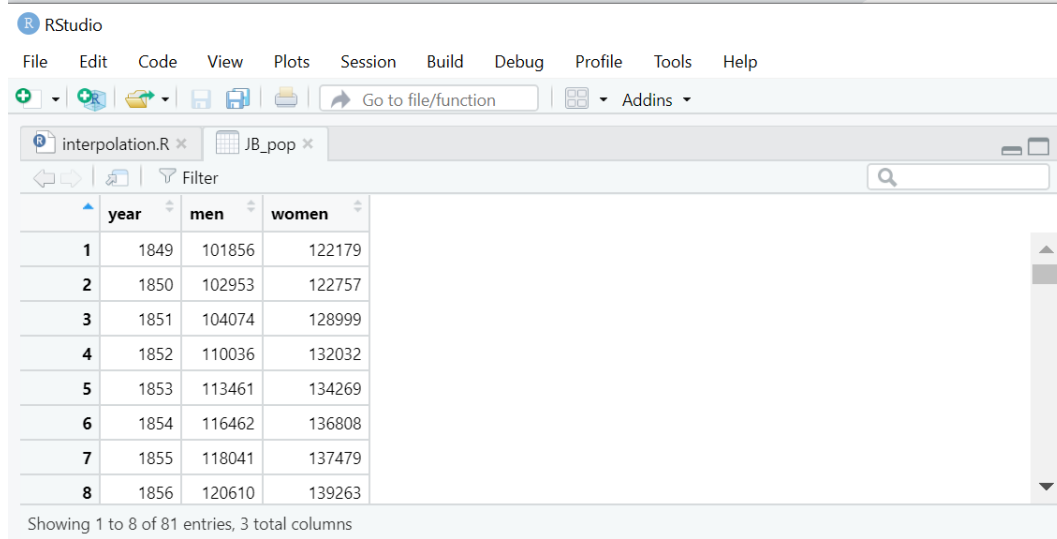
4 Comparison with data reported in the yearbooks

Now that we have obtained the annual mid-year population interpolation by gender in Section 2, we might want to know how the numbers relate to the actual total population reported in the *Statistische Jaarboeken* which provide total population information by gender but not by age-group. However, before being able to do that we have to realize that our data are not directly comparable because the yearbooks present the year-end population for each year. We have two options. First, we can redo our interpolation shown in Section 2 with a new *datesOut* vector with year-end dates instead of mid-year dates and compare those interpolated numbers with the population in the yearbooks. Second, we can interpolate the yearbook population numbers to mid-year and compare those with our interpolated data obtained earlier. I will show the first method in this tutorial.

First, let us familiarize with data from the yearbooks. An excerpt of the yearbook population dataset by gender (without age groups) is shown in Figure 8.

```
1 # importing Yearbook population data
2 JB_pop <- read_excel("JB_pop.xlsx")
```

Figure 8: Total population from the yearbooks by gender - sample



	year	men	women
1	1849	101856	122179
2	1850	102953	122757
3	1851	104074	128999
4	1852	110036	132032
5	1853	113461	134269
6	1854	116462	136808
7	1855	118041	137479
8	1856	120610	139263

Showing 1 to 8 of 81 entries, 3 total columns

In order to calculate the end-year population so that it is comparable with the yearbooks, we define a new *datesOut* vector and redo the interpolation.

```
1 datesOut_yearend <- c("1850-12-31", "1851-12-31", "1852-12-31",
2 "1853-12-31", "1854-12-31", "1855-12-31", "1856-12-31", "1857-12-31",
3 "1858-12-31", "1859-12-31", "1860-12-31", "1861-12-31", "1862-12-31",
4 "1863-12-31", "1864-12-31", "1865-12-31", "1866-12-31", "1867-12-31",
5 "1868-12-31", "1869-12-31", "1870-12-31", "1871-12-31", "1872-12-31",
6 "1873-12-31", "1874-12-31", "1875-12-31", "1876-12-31", "1877-12-31",
7 "1878-12-31", "1879-12-31", "1880-12-31", "1881-12-31", "1882-12-31",
8 "1883-12-31", "1884-12-31", "1885-12-31", "1886-12-31", "1887-12-31",
9 "1888-12-31", "1889-12-31", "1890-12-31", "1891-12-31", "1892-12-31",
10 "1893-12-31", "1894-12-31", "1895-12-31", "1896-12-31", "1897-12-31",
11 "1898-12-31", "1899-12-31", "1900-12-31", "1901-12-31", "1902-12-31",
12 "1903-12-31", "1904-12-31", "1905-12-31", "1906-12-31", "1907-12-31",
13 "1908-12-31", "1909-12-31", "1910-12-31", "1911-12-31", "1912-12-31",
14 "1913-12-31", "1914-12-31", "1915-12-31", "1916-12-31", "1917-12-31",
```

```

15 "1918-12-31", "1919-12-31", "1920-12-31", "1921-12-31", "1922-12-31",
16 "1923-12-31", "1924-12-31", "1925-12-31", "1926-12-31", "1927-12-31",
17 "1928-12-31", "1929-12-31")

1 # LINEAR
2 # men
3 men_linear_yearend <- interp(men_limited, datesIn, datesOut_yearend,
4 "linear", rule = 2)
5 men_linear_yearend <- round(men_linear_yearend, 0)
6 colnames(men_linear_yearend) <- seq(from=1850, to=1929, by=1)
7 men_linear_yearend <- cbind(age, men_linear_yearend)
8 men_linear_yearend <- as.data.frame(men_linear_yearend)
9
10 # women
11 women_linear_yearend <- interp(women_limited, datesIn, datesOut_yearend,
12 "linear", rule = 2)
13 women_linear_yearend <- round(women_linear_yearend, 0)
14 colnames(women_linear_yearend) <- seq(from=1850, to=1929, by=1)
15 women_linear_yearend <- cbind(age, women_linear_yearend)
16 women_linear_yearend <- as.data.frame(women_linear_yearend)
17
18 # EXPONENTIAL
19 # men
20 men_exponential_yearend <- interp(men_limited, datesIn, datesOut_yearend,
21 "exponential", rule = 2)
22 men_exponential_yearend <- round(men_exponential_yearend, 0)
23 colnames(men_exponential_yearend) <- seq(from=1850, to=1929, by=1)
24 men_exponential_yearend <- cbind(age, men_exponential_yearend)
25 men_exponential_yearend <- as.data.frame(men_exponential_yearend)
26
27 # women
28 women_exponential_yearend <- interp(women_limited, datesIn,
29 datesOut_yearend, "exponential", rule = 2)
30 women_exponential_yearend <- round(women_exponential_yearend, 0)
31 colnames(women_exponential_yearend) <- seq(from=1850, to=1929, by=1)
32 women_exponential_yearend <- cbind(age, women_exponential_yearend)
33 women_exponential_yearend <- as.data.frame(women_exponential_yearend)

```

The main goal here is to create a graph which shows the difference between the interpolated and yearbook population by gender and by year. For this, we have to construct a table that includes all the necessary information. For the graphs we use the *ggplot2* package which prefers tables in the long format. Preparing such a table requires various steps.

Step 1: Calculate the total sum of interpolated population by year

The codes below generate four numerical vectors with the total annual sum of age group populations by gender and interpolation method. Note that the first column of the underlying tables which contain the age group information is not taken into account. Each vector has 80 elements corresponding with the number of years between 1850 and 1929.

```

1 men_lin_yearend <- colSums(men_linear_yearend[, -1])
2 women_lin_yearend <- colSums(women_linear_yearend[, -1])
3 men_exp_yearend <- colSums(men_exponential_yearend[, -1])
4 women_exp_yearend <- colSums(women_exponential_yearend[, -1])

```

Step 2: Compile all data in a single table, calculate the difference and rearrange the table in a long format

We create a table, *comparison_gender*, which columns contain the actual and interpolated

population and the differences between those. This table serves as the main underlying table for the graph. Please note, that our interpolated data start in 1850 while the yearbook population data start in 1849, which we take care of in the first line of the code below.

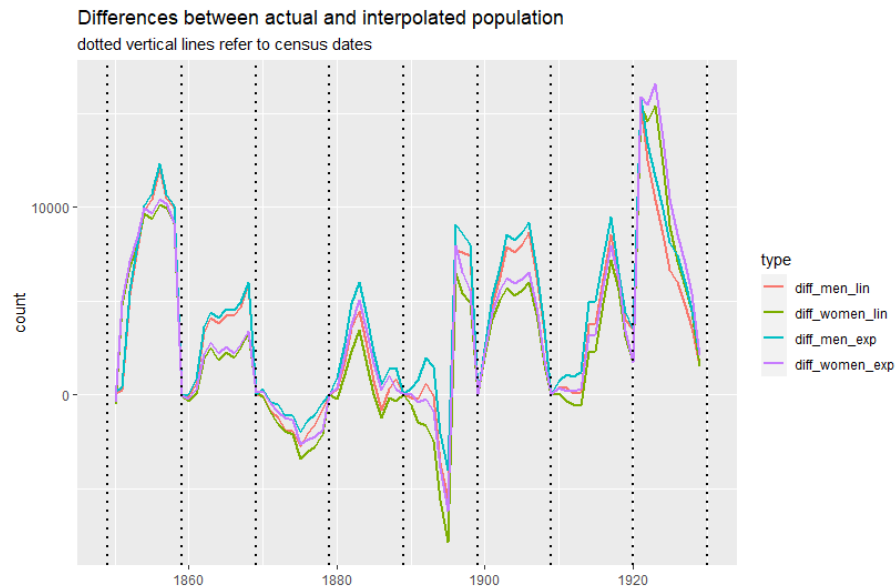
```

1 # wide table
2 ## compiling the data into a single dataset
3 comparison_gender <- JB_pop[-1,]
4 comparison_gender$men_lin <- men_lin_yearend
5 comparison_gender$women_lin <- women_lin_yearend
6 comparison_gender$men_exp <- men_exp_yearend
7 comparison_gender$women_exp <- women_exp_yearend
8
9 ## calculating the difference between the actual and interpolated data
10 comparison_gender$diff_men_lin <- comparison_gender$men-comparison_gender$men_lin
11 comparison_gender$diff_women_lin <- comparison_gender$women-comparison_gender$women_lin
12 comparison_gender$diff_men_exp <- comparison_gender$men-comparison_gender$men_exp
13 comparison_gender$diff_women_exp <- comparison_gender$women-comparison_gender$women_exp
14
15 ### plotting differences
16
17 diff <- subset(comparison_gender, select=c("year", "diff_men_lin", "diff_women_lin", "diff_
18   men_exp", "diff_women_exp"))
19 diff <- melt(diff, id.vars = "year",
20   measure.vars = c("diff_men_lin", "diff_women_lin", "diff_men_exp", "diff_women_exp"),
21   variable.name = "type",
22   value.name = "pop")
23
24 figure9 <- ggplot(diff, aes(x=year, y=pop, group=type, color=type)) +
25   geom_line(size=1) +
26   ggtitle("Differences between actual and interpolated population") +
27   ylab("count")+
28   xlab("")
29 figure9

```

Figure 9 shows the difference between the yearbook population and the interpolated population by method and gender. The vertical dotted lines indicate census years. The total population obtained by the interpolation method is most often below the yearbook numbers and the difference is the largest in the middle of the between-census periods. The smallest difference between the two sources is found in the census years.

Figure 9: The differences between the actual and interpolated population



The likely reason for this peculiar pattern is that the yearbooks probably report the population in the between-census years based on statistics on births, deaths, in- and outmigration from the city, and the population counts obtained with this method is much higher than the number we would get with actually counting the population at the end of the year (which is the census method). Thus, it is logical to use the interpolated population numbers for calculating various rates instead of the population numbers reported in the yearbooks. The researcher should him-or herself decide which data to use or perform data adjustments if possible.

References

- Riffe, T., Aburto, J., Alexander, M., Fennell, S., Kashnitsky, I., Pascariu, M., and Gerland, P. (2019). DemoTools: An R package of tools for aggregate demographic analysis. URL: <https://github.com/timriffe/DemoTools/>.
- van Leeuwen, M. H. and Oeppen, J. E. (1993). Reconstructing the demographic regime of Amsterdam 1681–1920. *Economic and Social History in the Netherlands*, 5:61–102.