Data visualization

LIMONIER Joris DE VILLIERS Marlize BIRBIRI Ufuk Cem

January 12, 2022

Contents

1 Sankey diagram by Joris LIMONIER

1.1 Introduction

1.1.1 Get the project

My submission for the project of data visualization consists of a Sankey diagram. It is available here [1] and should be viewed in the browser (tested on Chrome), on a computer. The code for the project is posted on GitHub.

1.1.2 Background on Sankey diagrams

Sankey diagrams are defined as follows: "Sankey diagrams are a type of flow diagram in which the width of the arrows is proportional to the flow rate." [1] In our case, we have four columns that are linked by flows, representing the number of albums or the number of songs.

1.1.3 Code structure

The diagram was created with the following structure:

- Python [2] for data pre-processing (cleaning, formatting, writing).
- CSV for the data relative to each passage from one column to another (4 columns, therefore 3 CSV files).
- JSON to group the CSV data and put them in a format that can be fed to the D3.js library.
- HTML for the website structure.
- CSS for the website styling.
- JavaScript (Vanilla) to perform last-minute grouping and filter-relative updates.
- D3.js [3] for the interactive data visualization.

Let us first detail the data preprocessing part.

1.2 Data pre-processing

1.2.1 Raw data

We grab the following raw data from the CSV Wasabi datasets that were downloaded.

- Album field
  - _id (album id)
  - id_artist
  - genre
- Artist field
  - _id (artist id)
  - type (*e.g.* "Person", "Orchestra", "Group", ...)
  - gender (male, female, unknown)
- Song field
  - id_album

[1] If the link is dead, go to https://jorislimonier.github.io/, navigate to "Projects", then look for "Collaborative Data Visualization"

1.2.2 Clean the data

The number of NaN values varies significantly form column to column. The Id columns don't have any but some columns have many, making them sometimes tedious to work with. Overall however, the dataset is pretty clean. Barely a few Id's are misformatted and some genres have unexpected characters.

Let us call "column transfer" (CT) the passage from one column to another. This is what we feed into D3.js and this is the part with the most impact on the diagram.
We clean the data in *sankey.py*, which contains one class per CT :

- *TypeGender* : goes from the artist type to the gender of the artist.
- *GenderAlbums* : goes from the gender of the artist to the number of albums produced.
- *AlbumsSongs* : goes from the number of albums produced to the average number of songs per album.

The same file also contains the following class:

- *Sankey* : contains several functions to go from the individual CT classes to writing the data in the final format.

Each of the CT classes writes a CSV file. Usually, Sankey diagrams have three columns:

- source : the origin node for this flow
- target : the target node for this flow
- value : the flow quantity (*i.e.* the number of elements that go from a given source to a given target)

The *TypeGender* CT does indeed have these three columns. The *GenderAlbums* and *AlbumsSongs* CT 's on the other hand, are composed of a fourth column:

- genre : the genre of the album

This genre column allows to perform the filter operation.

1.2.3 Transfer the data

The *Sankey* class has a $write\_final\_data$ method that takes the three CSV's and writes the format in a JSON file (sankey-genre.json), in the format demanded by D3.js. The format is as follows:

- nodes:
  - index
  - name
- links:
  - source
  - target
  - genre
  - value

where the set of the node indices is simply a bijection between the unique node names and $\mathbb{N}_{\geq 0}$.

1.2.4 Set up filtering

Filtering is performed in *sankey-filter.js*. We allow for a filtering that only takes place for the two last *CT* 's (*i.e. GenderAlbums* and *AlbumsSongs*), because the *TypeGender CT* concerns artists, not albums (and obviously, artists don't have a genre).
We use JavaScript for the filtering part because we need to update the graph when genres are added/removed from the filter list. On each change of the selection dropbox (using an *eventListener*), we go through sankey-genre.json and check for each element in the "links" list if it is in the list of genres selected by the user. If so, we group it with all other elements going from the same source node to the same target node and sum their values.

1.2.5 Group genres

The grouping of genres is performed in *sankey.py*. It contains a dictionary-like object that is of the following structure:

- keys: the new group name.
- values: a list of strings. If the old genre contains one of these strings, it will be assigned to the new group which is named after the *key*.

1.3 Data Visualization

1.3.1 General intention

This visualization tries to stay close to Schneiderman's mantra: "Overview, Zoom and Filter on demand" [4]. It appeared however that the zoom feature didn't really find any meaningful usecase for this Sankey diagram, which is why it has been omitted.
Table 1 details the actions available for each of the general task categories.

	User task	Details
1.	Overview	The artists gender per artist type The number of albums per artist gender The number of songs-per-album per number of albums
2.	Selecting	Drag and drop nodes to rearrange them Hover over the links to flow value, source and target
3.	Filter	Only show data for one or more album genres

Table 1: User tasks for the Sankey Diagram.

1.3.2 Overview

Figure 1 shows an overview of what the default view looks like. The boxes on top show column names. They could probably be inferred from the node labels, but it is meant to facilitate the User Experience.

1.3.3 Select

If the user wishes to rearrange the nodes from each column, they can drag and drop them. This allows to isolate a single segment of the data for a temporary visualization, *e.g.* if someone wants to see how many female made 3 vs 4 albums with 2 songs/album. The other informations can be dragged to the bottom to disregard them. Figure 2 shows such an operation. Note that this is more of a "quick-and-dirty" way of visualizing that more females made 3 albums than 4 albums, rather than a real use case for most of users.

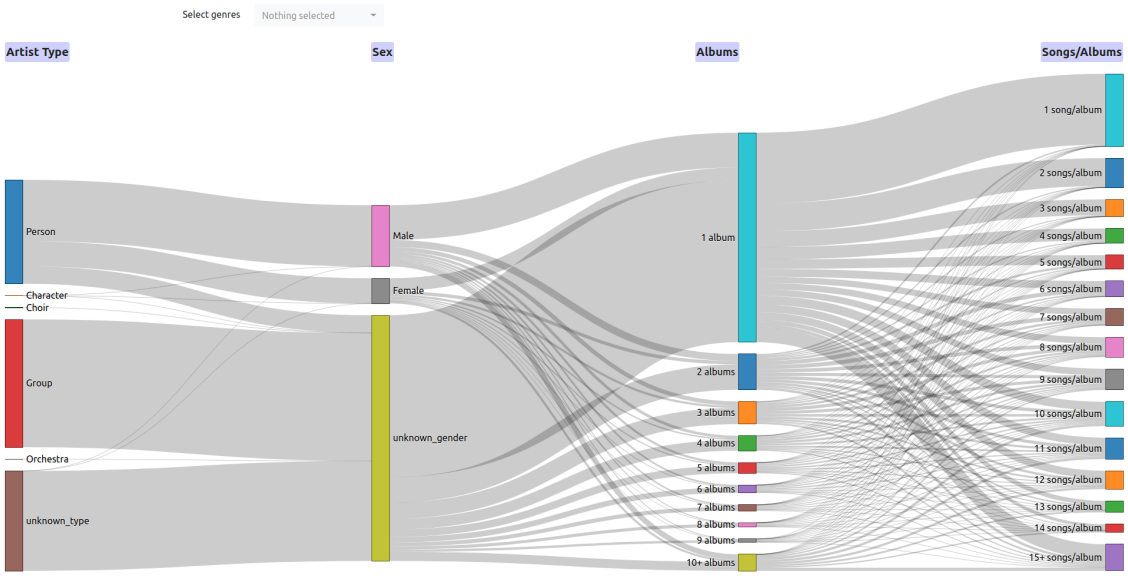

Figure 1: Overview of the default Sankey diagram.

Moreover, if the user wants to have more information on one specific link, *e.g.* knowing how many artists of unknown gender produced 3 albums, this can be done by hovering over the link. Then a tooltip appears (see Figure 3), showing complementary information in the following format:

$$\text{source} \rightarrow \text{target}$$
$$n \text{ occurences}$$

where n is the link value from the given source to the given target.

1.3.4 Filter

Subsequently, the user can choose to show the data only for a few genres. The selection dropdown is displayed in Figure 4.

When the user clicks to add/remove a genre from the list, the diagram is recomputed and redisplayed on the page. The width of the links update accordingly, as shown in Figure 5. As mentioned previously, only albums have a genre (artists don't), which is why the left-most CT remains untouched after filtering.

1.4 User test

1.4.1 Perform tasks

The user test was performed on 3 testers who agreed that their information is being gathered. They were aged 22-24, with a Bachelor's degree in a scientific discipline.
The users were asked the following questions:

1. Task 1
   (a) How many men made 5 albums.
   (b) On a scale from 1 to 5 (1 means very easy, 5 means very difficult), how hard was that task?
2. Task 2

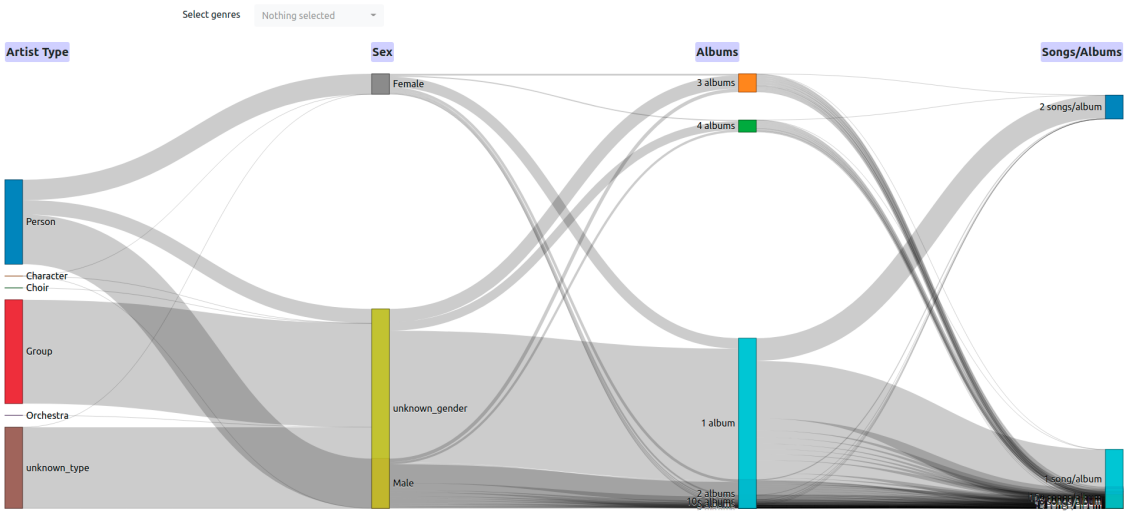

Figure 2: Display of the "quick-and-dirty" isolation

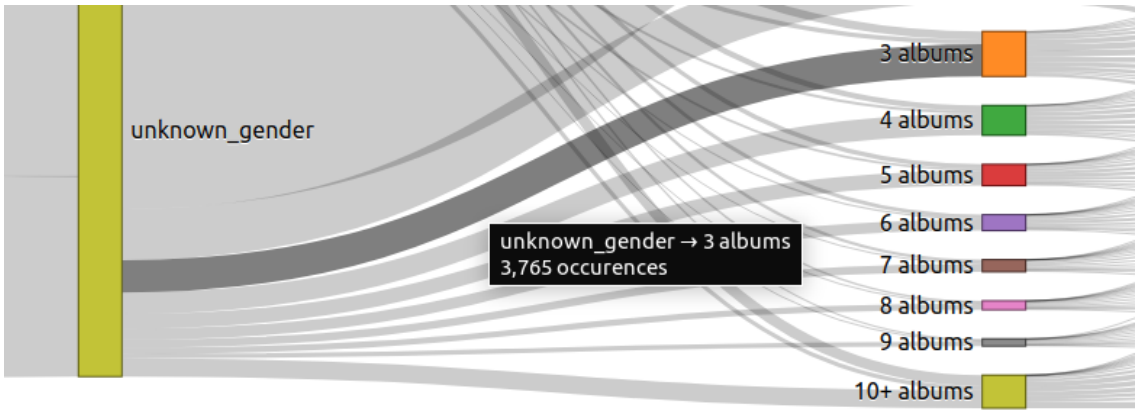

Figure 3: Display of the tooltip on hover

   (a) How many rap solo artists are women.

   (b) On a scale from 1 to 5 (1 means very easy, 5 means very difficult), how hard was that task?

3. Task 3

   (a) Can you tell how many solo artists made 3 albums? Why/Why not?

   (b) On a scale from 1 to 5 (1 means very easy, 5 means very difficult), how hard was that task?

Task 1 and 2 were performed successfully by the participants, who found it rather easy to complete (one of them mentions that some time was needed to get familiar with the diagram, since he was not familiar with them).
The third task is a trap since there is no CT that allows to give this information. 2 participants identified that this was impossible, the third one said that the were unable to perform the task (therefore they didn't understand that the task was undoable).

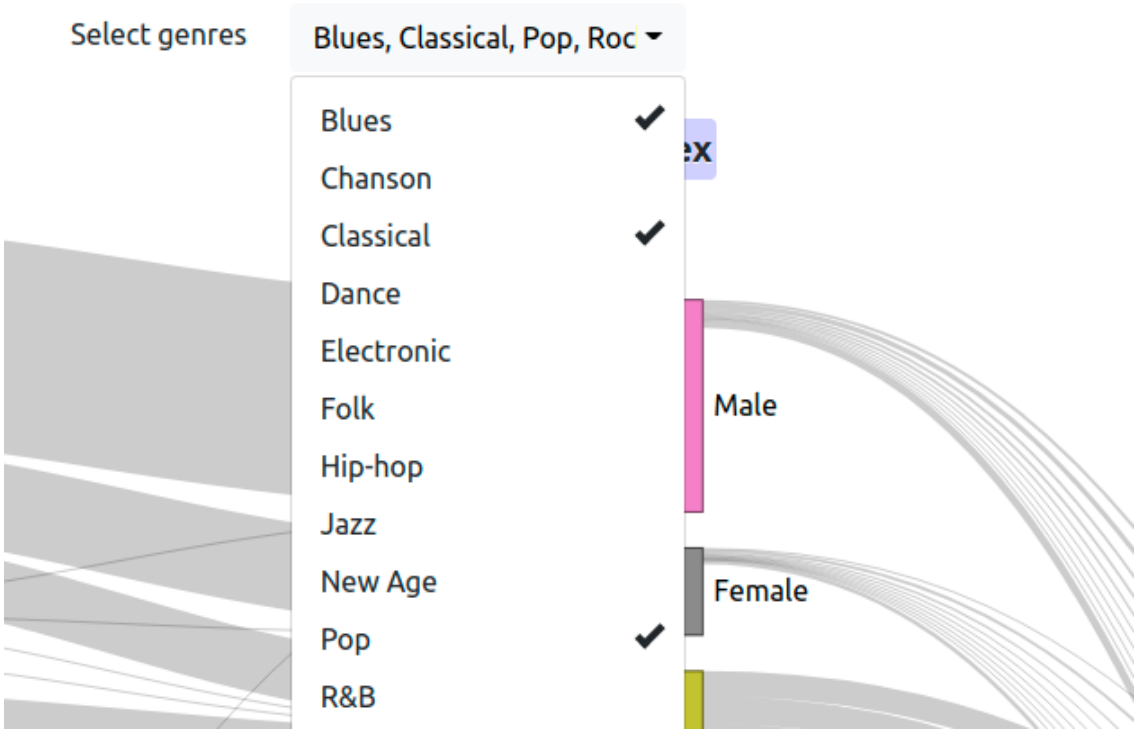

Figure 4: Display of the filter options where “Blues”, “Classical” and “Pop” are selected

1.4.2 Debriefing

The users reported that overall, they enjoyed interacting this visualization (3/3) and that they would recommend it to a friend (3/3). Their least favorite part of the data was that the “unkown_genre” genre is over-represented. The way to improve this issue is either to reduce the number of missing values in the data set (but this is a data collection matter, which is not my main concern), or to better group the genres. This can be achieved by two ways: grouping more genres under the ones already present, and adding more grouped genres.
One user mentioned that they would have liked to see smooth animations when selecting other genres. This is probably doable through better knowledge of D3.js, but since this was my very first experience with JavaScript at all and since the time was constrained, the final version of my Sankey diagram doesn’t contain transitions.

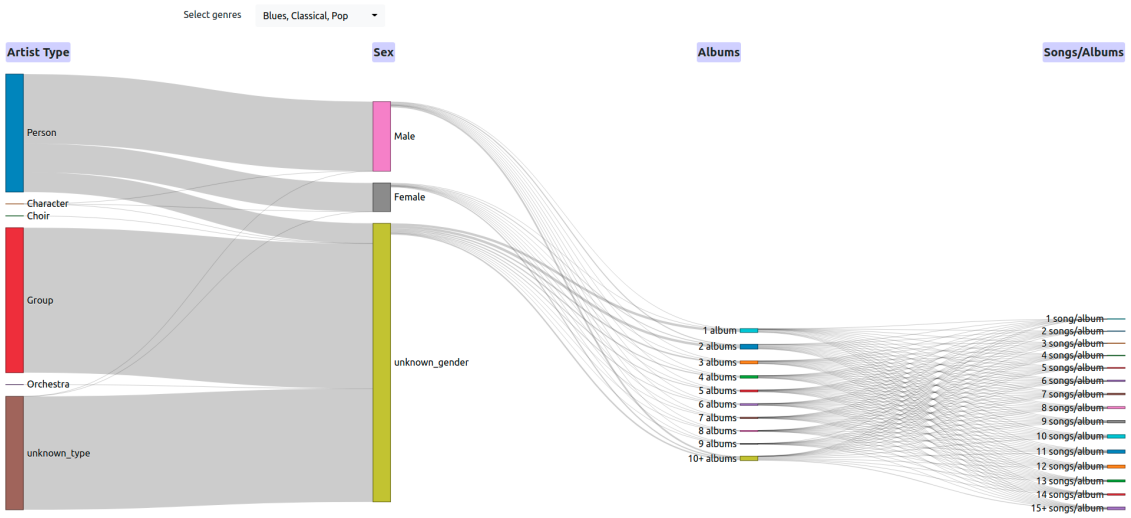

Figure 5: Display of the diagram where “Blues”, “Classical” and “Pop” are selected

2 Bubble map by Marlize DE VILLIERS

A bubble map was chosen for the vizualisation technique. The vizualisation shows the number of albums released and/or awards received per country in a given period or year. The users of the visualization are people who are interested in seeing how many albums were released in different countries over time and how many awards the albums received over time. It could also be useful for up-and-coming artists who want to maximise their chance of receiving an award for their album based on which locations publish more albums and receive more awards.

2.1 The user tasks

The tasks that the user can perform with the vizualisation are described in Table 2.

Table 2: A list of user tasks.

	User task	Details
1.	Overview	An overview showing the bubbles of albums released and/or awards received in all of the countries over all of the years.
2.	Selecting	Hover over bubbles to see more detailed information.
3.	Filter	Only show countries that have released more than a certain number of albums. Show only albums released or only awards received for a specific period. Show only albums released or only awards received. Show or hide the incomplete information in the dataset. Show only certain continents' data.
4.	Zoom	Zoom into a specific region or country.

2.2 The raw attributes needed from the WASABI dataset

The following attributes were necessary to produce the vizualisation:

- From the album field
  - _id (album id)
  - id_artist
  - publicationDate
- From the song field
  - id_album
  - award
- From the artist field
  - _id (artist id)
  - location: country

2.3 A description of processing steps required

The main objectives of the data cleaning and pre-processing was the following:

1. To get the location where each album was published.
2. To get the release date of each album.

3. To determine if any songs in each album received an award.

But to do this, many steps were necessary. These steps will briefly be described in the following paragraphs.

After the attributes that were required were identified, they were extracted from the various datasets that make up the WASABI dataset. The attributes were compiled into three "new" datasets — *albums*, *artists*, and *awards* — to make the handling of the data a bit easier, since it takes very long to load the entire dataset each time.

2.3.1 Cleaning the album data

To clean the album data, the "object()" around the album_id and artist_id was removed. This was done because in some cases the ID did not have the "object()" characters around it, which would make it very difficult to match and merge the data from the three different datasets, because the merge will be done on the ID fields. Then, some of the publication dates contained more characters than just the year that the album was released. In this case, the publication year was extracted using string comprehension techniques. There were also some entries with missing dates. In these cases, the publication date was set to "Unknown", since it is not a good idea to exclude incomplete data from the final datasets.

2.3.2 Cleaning the songs data

For the songs dataset, the "object()" around the album_id for each song was removed and then a boolean variable was set to true to indicate if a song received an award. Then the songs dataset was grouped by the album_id, and the number of "True's" for each album was counted to get the number of awards each album received. If there were only "False's" in the dataset, the number of awards for that album was set to 0. The "songs grouped by album_id" dataset was then merged with the album dataset, again using the album_id. This generated some missing values, because not all of the albums were in the songs dataset. Because of this, we do not know if those specific albums received awards or not, so these albums' number of awards were set to "Unknown".

2.3.3 Cleaning the artist data

The data in the artist field was very messy. Some of the country locations were actually cities and vice versa, and some locations were actually states in the USA, or regions in certain countries. There were also locations that were spelled in the language of the country and not in English. All of these issues had to be handled. To prevent losing any further data in the already sparse dataset, both the country and city fields were checked and compared to a list of all of the countries and cities in the world as well as all of the states in the USA. The comparison was done as follows:

1. If the country field was a country in the list of countries, it was left as-is.

2. If the country field was actually a city, the corresponding country was found in the list with each city and its country.

3. If there was something in the country field that did not match a country or city, it was compared to the list of states in the USA, and if it matched, the location was set to "United States of America".

4. If there was something in the country field that did not match anything, the content was added to a list of "locations not found".

More or less the same steps were repeated with the city field.

After the entire dataset was compared, the locations that were not found were inspected. It included things like "Wien" which is Vienna in Austria, "Veneto" which is a region in Italy etc. Since this list was not too extensive, each location's country was searched for and then the location

and it's corresponding country were added to the list of cities with their corresponding countries. The entire process was then repeated to get as many useful locations as possible. At the end of this process, if there was still no information on the location, the location was set to "Unknown". Lastly, similar to the albums data, the "object()" around the artist_id was removed, and this dataset was merged into the dataset containing the cleaned albums and songs data.

2.3.4 Creating the dataset used in the vizualisation

After the data were cleaned, the dataset that was created by merging the artists, albums and songs datasets, contained the album- and artist_id's, the album's publication date and location, and the number of awards each album received. From this dataset, the publication date, location and number of awards were extracted. This dataset was then grouped by the publication date and location to get the total number of awards received in each year in each location. A second grouping was then made on the same dataset, but this time, counting the number of observations in each location-date-group. This gives the number of albums released for each year and location. These two new datasets were then merged on the location and date. The new dataset contained the location, date, number of albums released, and the number of awards received.

Then, the latitude and longitude of each country's capital was required to place the bubbles on the map. This was done using a dataset containing all of the countries in the world, along with their capitals and the coordinates of the capital. The cleaned data was given three new columns, "latitude", "longitude", and "continent". The continent was used to color the bubbles based on the continent. Each country in the cleaned dataset was then matched to the same country in the dataset described above, and the new columns were then populated correspondingly. The resulting dataset is the final dataset that was used in the vizualisation.

2.4 The vizualisation

The first thing that the user will see is the *Introduction* panel. Figure 6 shows what this panel looks like. It contains some information about the vizualisation, explains how to use it, and some general

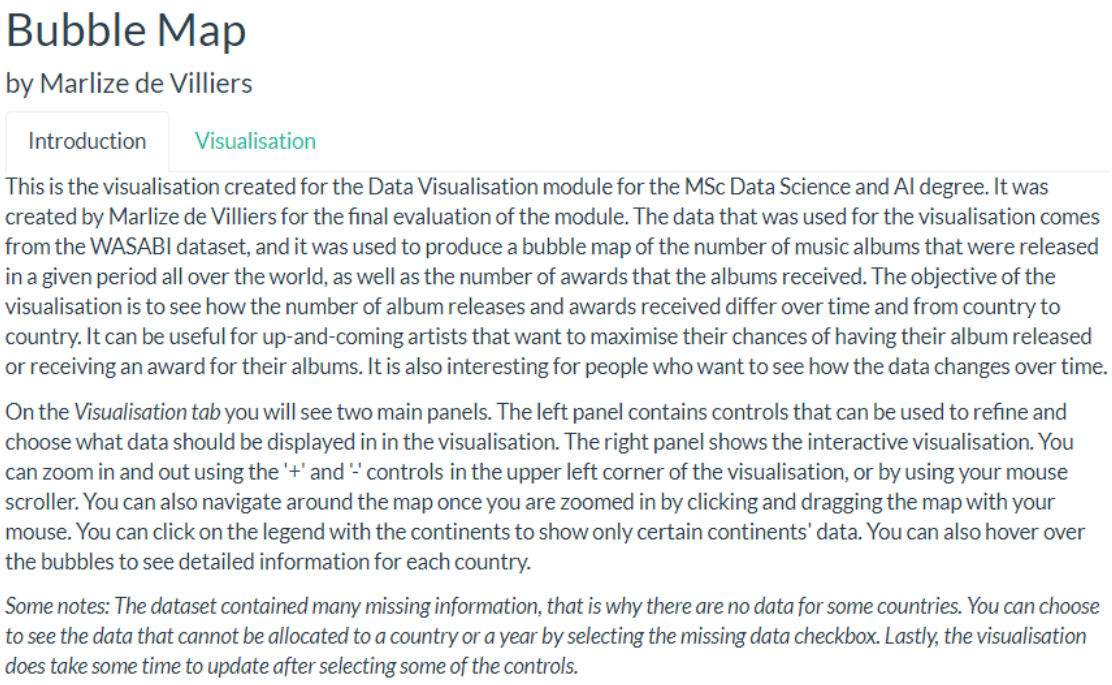

Figure 6: The first page of the vizualisation.

notes about the vizualisation.

On the *vizualisation* panel, the user will see the vizualisation controls on the left and the vizualisation on the right. Figure 7 shows what the controls look like, and Figure 8 shows what the vizualisation looks like. The user can choose any combination of the controls, and the vizualisation

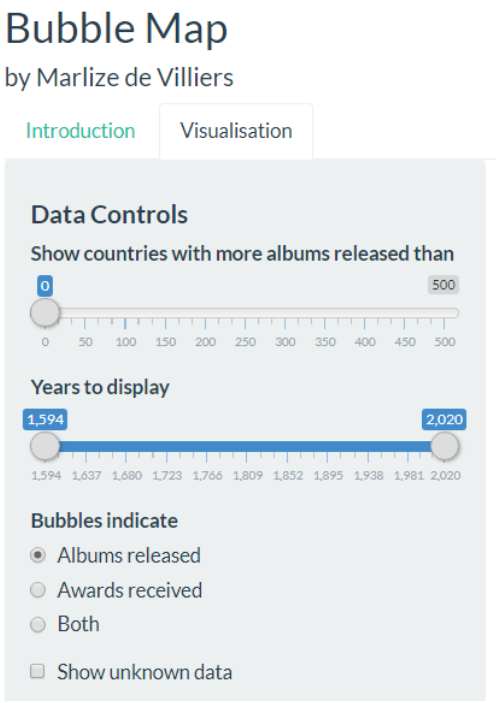

Figure 7: The controls of the vizualisation.

updates after each change. If a particular controls combination results in no data, only the map (without any bubbles) will be shown. But if the control combination results in data, something similar to Figure 8 will be displayed.

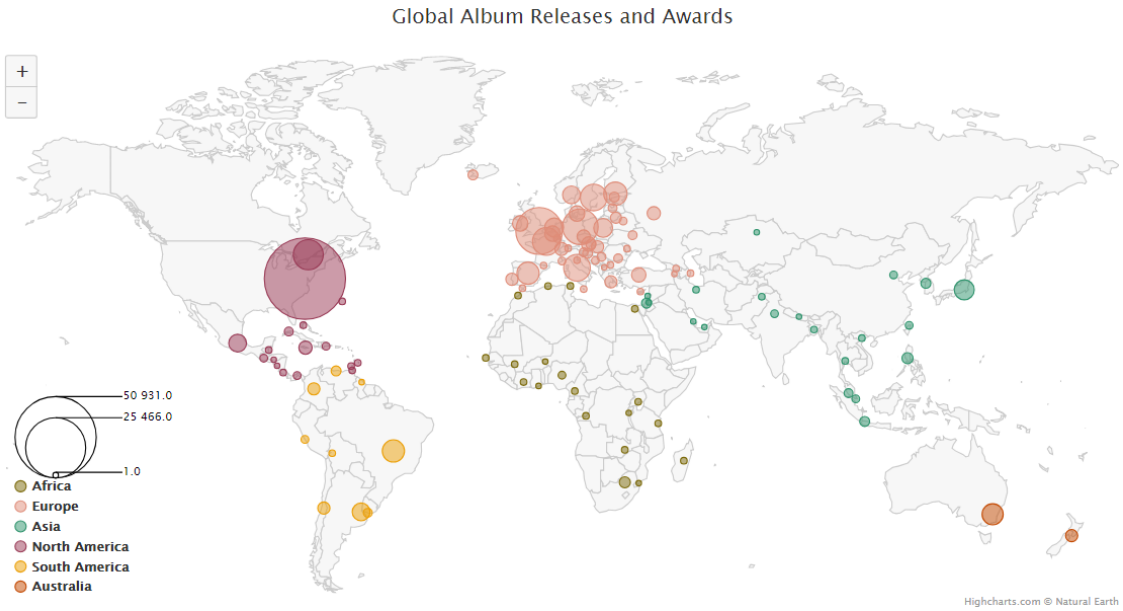

Figure 8: The vizualisation.

If the user chooses to see the "Unknown" data, they will see the information in a box below the map, as is shown in Figure 9.

The user can also zoom into any part of the map and navigate around the map once they have zoomed in. This is particularly useful to see the data in the European region better, since the

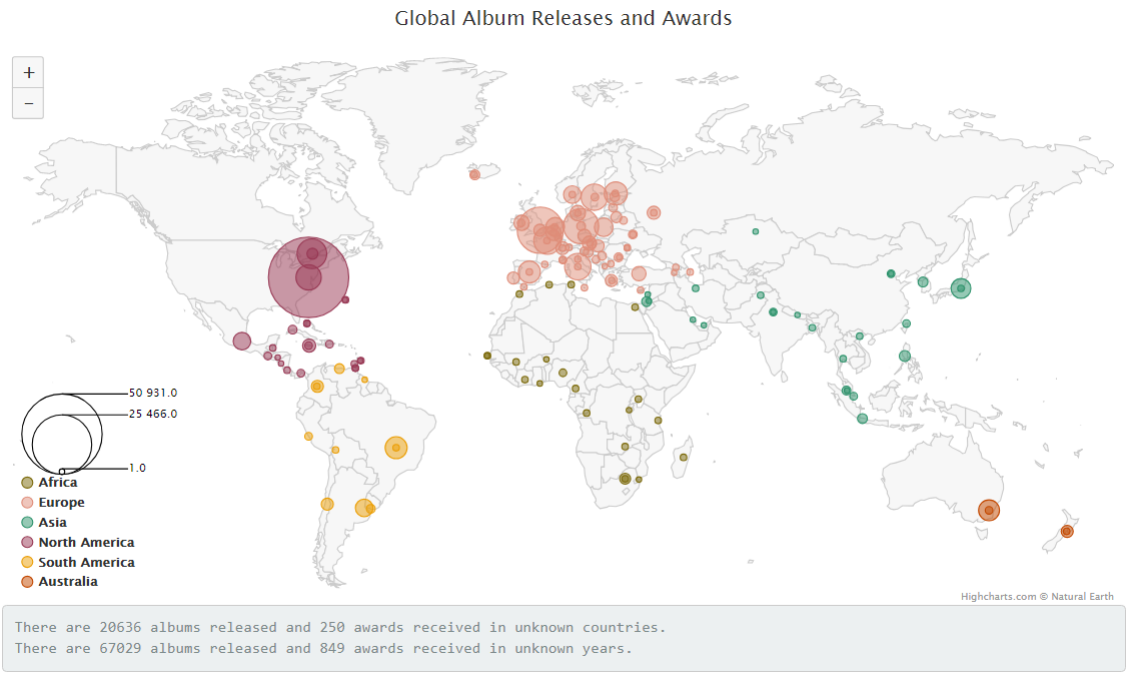

Figure 9: The vizualisation with unknown data.

bubbles are very densely spaced there. Users can also hover over bubbles to get more information about the particular country in the form of a tool-tip. This is shown in Figure 10.

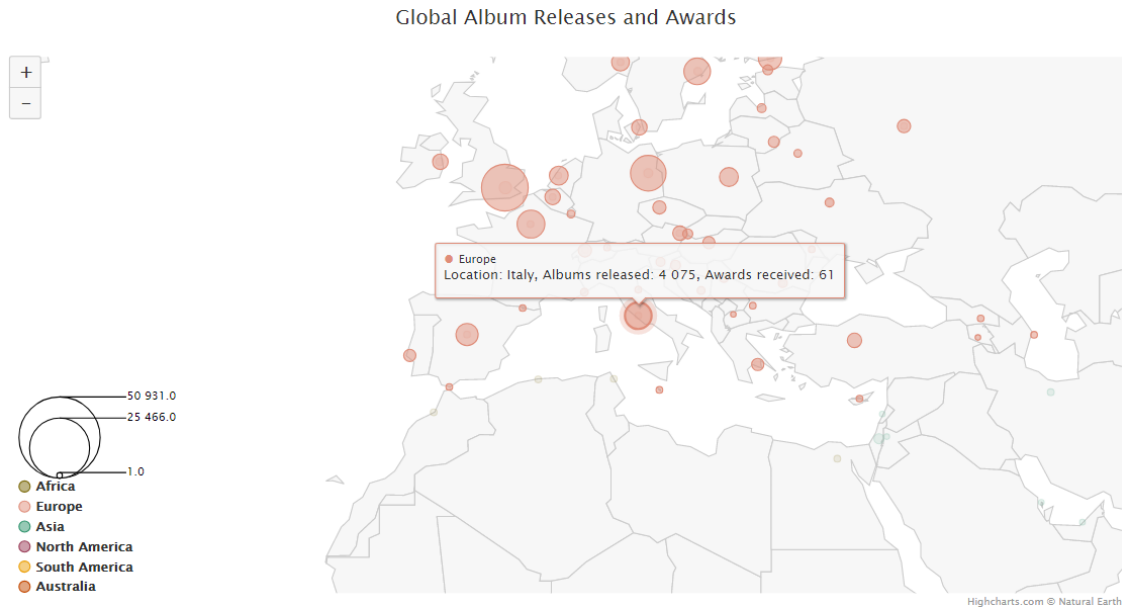

Figure 10: Zooming in on vizualisation.

The latest version of the interactive vizualisation is deployed here:
https://marlize-de-villiers.shinyapps.io/Data-visualisation-Bubblemap/.

2.5 User testing

A user testing protocol was set up and then two user tests were performed. The results of the tests are shown in Figure 11. It should be noted that the user test was completed before the vizualisation was 100% finalised, so the feedback was taken into consideration while the bubble map was finalised.

The main issues that were detected were that the users would like more colors, and that it

Timestamp	2021/12/08 11:09:14 am CET	2021/12/08 11:25:45 am CET
Do you consent to your information being gathered?	Yes	Yes
Demographics question 1: How old are you?	22	22
Demographics question 2: How often do you listen to music?	2hrs	3hrs
Task 1: How many albums were released in France? (Answer)	5487	5487
Task 1: How many albums were released in France? (Time)	17.51	43.3
Task 1: How many albums were released in France? (Difficulty)	1	4
Task 2: When you show only the data for countries that have released more than 60 albums, is there a bubble for Monaco? (Answer)	No	No
Task 2: When you show only the data for countries that have released more than 60 albums, is there a bubble for Monaco? (Time)	39.95	13.77
Task 2: When you show only the data for countries that have released more than 60 albums, is there a bubble for Monaco? (Difficulty)	2	5
Task 3: How many awards were received in Canada? (Answer)	Na	7341
Task 3: How many awards were received in Canada? (Time)	37.98	20.56
Task 3: How many awards were received in Canada? (Difficulty)	4	5
Task 4: How many albums were released in unknown locations? (Answer)	49635	49653
Task 4: How many albums were released in unknown locations? (Time)	9.9	12.53
Task 4: How many albums were released in unknown locations? (Difficulty)	5	5
De-briefing Question 1: Name three things that you like about the visualisation?	Easy to understand easy to navigate functionality, zoom	Zoom, slider, unknown info
De-briefing Question 2: Name three things that you do not like about the visualisation?	Bubbles too close, colors per continent,	Colors
End-of-test Question 3: Would you add or change anything to the visualisation? If yes, what specifically?	Colors, bigger zoom button	Colors
End-of-test Question 4: Would you tell your friends about this visualisation?	Yes	Yes

Figure 11: User test results of the bubblemap.

was not obvious that it is possible to zoom into the map. These issue of color was handled by coloring the bubbles according to the continent. The issue of the zooming was handled by giving a brief explanation of how the vizualisation works and what kind of actions can be performed in the introduction of the application.

3 Word Cloud by Cem BIRBIRI

3.1 Users Description

Users are people interested in lyrics of songs and genres. Lyrics are words used in songs. Users can visualize what kind of lyrics and how many of them are used in a specific genre with a word cloud. To do that, I used lyric summaries of the songs because complete lyrics are not publicly available, however lyric summaries of the songs are available. So as a summary, I grouped the songs by genre and took their lyrics summaries then visualize them with a word cloud. More detailed explanation is in the rest of the document.

3.2 Visual tasks and the visualization goals.

The visual tasks in Word Cloud are shown in the table below.

User task	Details
Explanation of word cloud	Give info about what is word cloud
Explanation of stopwords	Give info about what is stopwords
Explanation of the app	Give info about app and wasabi dataset
Choose a genre	Choose a genre to display its lyric
Remove stopwords	Removing the stopwords from the lyrics
Remove specific words	Removing words given by user
Show the count of the word	When user hoover on words on word cloud, show the count
Select years	Select a range of years to display lyrics in that range
Show lyrics that don't have publication date	Show lyrics that publication date is NA
Show count of songs	Display the number of songs in the given years
Choose word cloud theme	Select the word cloud visualization theme(Dark, Barbie,...)
Select shape	Select shape of the word cloud (diamond, circle, ...)
Select size	Select size of the word cloud

What are stopwords?
Stop words are basically a set of commonly used words in any language, not just in English. The reason why stop words are critical to many applications is that, if we remove the words that are very commonly used in a given language, we can focus on the important words instead.
Some examples of English stopwords:
i, me, my, myself, we, our, ours, ourselves, you, your, yours, they, what, which, who, whom, this, that, these, those, am, is, are, was, were, be, been, being, have, has, had, do, does, did, a, an, the, and, but, if, or, because, as, until, s, t, can, will, just, don, should, now, ...

3.3 Attributes from the WASABI dataset

The following attributes will be needed from the song field:

- Song field
  - _id (Song id)
  - genre (type:string)
  - summary (Summary of lyrics in a few lines, type:string)
  - publicationDate

3.4 Processing raw data

Data manipulation is done in the following order.

- Load and analyze data.
- Clustering the genres.
- Assign each song to a cluster of genre.
- Group songs by genre, clear lyrics, create csv files for each genre.

You can find the detailed explanation of these steps below.

3.4.1 Load and analyse the data

The data is loaded from wasabi web page and useful columns are selected such as song id, genre and publication date. The empty and not-given genres are detected and replaced by "Not specified". Same method is used for publication dates. The empty and not-given publication dates are detected and replaced by "0000".

Given publication dates are cropped and only the year is taken, month and day information is ignored. Example: 1998-06-22 becomes 1998.

In the wasabi dataset, one song can belong to multiple genres. In this situation, the genres are splitted. Example:
- ObjectId(5714dec325ac0d8aee38093d), "metal, punk, rock"

becomes:
- ObjectId(5714dec325ac0d8aee38093d), "metal"
- ObjectId(5714dec325ac0d8aee38093d), "punk"
- ObjectId(5714dec325ac0d8aee38093d), "rock"

where the ObjectId(...) is the song id.

3.4.2 Clustering the genres

There are many genres in wasabi datasets. Some genres are originated from other genres by years. For example, 'Skiffle' is originated from genre 'Folk', or 'Emo' is originated from 'Punk'. At this step, different genres are clustered according to their ancestors. For example, samba, bachata and salsa are grouped under the Latin music. In addition the Skiffle music genre is originated from folk music so it is grouped by folk music genre. Below, you can find the main genre and its cluster members. It is good to remember that a new genre can be a mixture of multiple genres. Here, clustering is made by taking into account the most similar genre.

Clusters of genres				
Rock	**Pop**	**Metal**	**Hip-Hop**	**Blues**
-Rock music	-Pop music	-Metal music	-Hip-hop music	-Blues music
-Surf music	-Wall of Sound	-Deathcore	-New jack swing	-Liedermacher
-Industrial music	-Beach music	-Deathgrind	-Miami bass	-Boogie-woogie
-British Invasion	-Vocal music		-Jumpstyle	-Bolero
-Shoegazing	-Music Hall		-Hands Up	-Tulsa Sound
-Minneapolis sound			-Hyphy	Singer-songwriter
-Neue-Deutsche Welle			-Lo-fi	
-Experimental music			-Crunk	
-Beat			-East Coast hip-hop	
-Dark cabaret			-Swing revival	
-Palm Desert Scene			-Breakbeat	
-Adult album alternative			-Yé-yé	
-Motown			-Police procedural	
			-Snap	

Dance	**Electronic**	**Jazz**	**Raggae**	**Country**
Dance music	Electronic music	Jazz music	Raggae	Country music
Disco	Big beat	Quiet storm	Dubstep	Honky-tonk
Hi-NRG	Grime	Afrobeat	Reggaestep	Music of Lubboc
Garage	Électronique	Dixieland	Hardstep	Texas
Speed garage	Musique-électronique	Stride	Ska	Music of Ireland
Freestyle	Remix	Screamo	Dub music	Western music
Bhangra		Tin Pan Alley	Oldschool jungle	Moombahton
Cabaret		Cumbia	Old-time	Kuduro
Low fidelity			Mento	
Sirtaki			Traditional black gospel	
Tropicália			Ragtime	

Funk	House	Rap	Folk	Soul	Punk
Funk music	House music	Rap music	Folk music	Soul music	Punk music
Show tune	Lullaby	Ballad	Skiffle	Stoner	Emo
World music	Teenage-tragedy song	Freestyle music	Tejano	Kwaito	Ragga
Go-go	Plunder-phonics		Baggy	Highlife	PBR
	Balada		Topical		Mariachi
	Gaana		Music of Scotland		Candombe
			Boogie		Junkanoo
			Ballet		

New Wave	Psychedelic	Christmas	Trance	Techno	Grunge
New wave	Psychedelic music	Christmas music	Trance music	Techno	Grunge
Cumbia			Contemporain	Downtempo	Post-grunge
New-age music					

Latin	Contemporary	Classical	Hardcore	Harmony
Latin music	Contemporary	Classical music	Hardcore music	Harmony music
Bachata	Madchester	Orchestra	California Sound	Doo-wop
Samba	Baroque	Crossover	Exotica	Drone
Salsa		Ambient		Worldbeat
Bossa nove		Patriotic		Schlager
Pasodoble		Piano		
Rumba				
Flamenco		Waltz		A cappella
Zumba.		Música sertaneja		Music of Italy

Instrumental	Poetry	Acoustic	Religious	Celtic	Comedy
Instrumental music	Poetry	Acoustic music	Religious music	Celtic music	Novelty music
Minimal	Ballade	Hymn			Parody music
Soundtrack	Chanson		Christian		Comedy music
March		Easy listening	Gospel		
Musique concrète		Bebop	Siren		
Kayōkyoku			Spoken		
Jimmy Buffett			Feminism		
Circus					
Calypso					
Doctor Who fandom					

3.4.3 Assign each song to a cluster of genre

With a for loop, every song is assigned to a cluster of genre. A new column is created for this assignment that is called 'grouped_genre'. This new column is used for grouping the songs by genre.

3.4.4 Group songs by genre, clear lyrics, create csv files for each genre.

Songs are grouped by genres. Then, songs are matched with corresponding lyrics. These lyrics are cleared before writing a file. Clearing is removing the punctuation and space. For example:
I will! becomes I will
Then, a .csv file is created for each genre which has lyrics and publication date. The data is ready for word cloud in visualization.

3.5 The visualization technique

I will use Word Cloud to represent words in each genre. The size of the word in the tag cloud will represent the frequency of the word i.e if a word is used a lot in lyrics, its size will be bigger than others. Lyrics are made of words so word cloud is a good choice to visualize lyrics to see how often the words are used.

3.6 A visual mapping of variables

Each genre will has its own word cloud since each genre has its own csv file. In this file there are two columns. The first one is the publication year of the song and the second one has the lyrics summaries of the song. The number of rows is the amount of songs belong to that genre. The Fig 12 shows the first three rows of the pop.csv file which has lyrics and publication date of songs for genre pop.

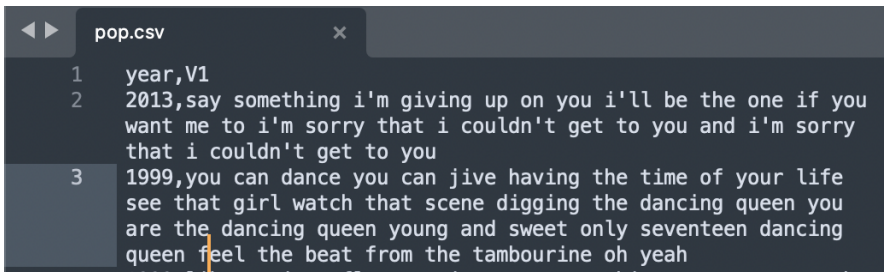

Figure 12: The first three rows of pop.csv file. This file includes the lyrics and publication years of the songs. Each row is a different song. The first column shows the publication year and named as 'year'. The second column shows the lyric summaries of the song and named as 'V1'.

Word Cloud is a vizualisation method that displays how frequently words appear in a given body of text, by making the size of each word proportional to its frequency. Colour used on Word Clouds is usually meaningless. I will not use colour, every word will have the same colour. The location of the word is also meaningless. Some words are written horizontally and some are written vertically which are random and do not mean anything. Only meaning feature in the word cloud is the size of the word.

The Fig13 shows the word cloud of genre pop in the Shiny app. The words are visualized proportional to their frequency. Users can see how many times a word is used by passing around the mouse on the words. According to figure, the word 'love' is used 587 times.

Figure 13: The word cloud representation of genre pop between the years 1912 and 2017.

3.7 Detailed explanation of the Shiny app

3.7.1 Before running app

1. You should open the Shiny app in full page in your computer screen. The world cloud fulfills the screen if you open the app full screen. However, if you open the app in a small screen and then enlarge it, the word cloud size(the black background at the beginning) will keep being small. To fit it the to screen you should reload the page.

2. After matching the songs with lyrics, there are 46615 songs that does not have any genre in the wasabi dataset. These songs are represented under the genre "Not Specified". Not Specified genre is also displayed in the Shiny app. You can find it in at the bottom of the genre list on slide-bar panel. Since there are many songs in this genre, it takes some time to run the code and display the world cloud.

The Fig14 shows the visualization of Shiny app options. Explanation of the Shiny app and usage are listed below.

3.7.2 On the slide-bar panel:

- Choose type of genres to see the lyrics on the word cloud.

- Select a range of publication years. Publication years on the slide-bar change when the user selects different genre.

- There are some songs that the publication date is not given in the dataset. So,if the user selects the checkbox button of 'Only shows songs that publication date is not specified in dataset?', the word cloud shows only the songs that do not have a publication date. Slide bar of the publication years does not work with this option.

- Remove stopwords.

- Remove specific words in the word cloud (maximum 10 specific words).

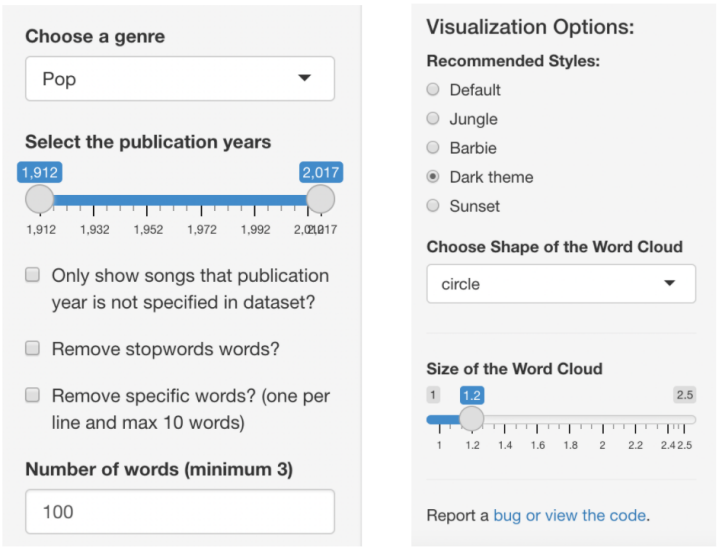

Figure 14: Left: User options for song lyrics, publication year, stopwords and the number of words displayed in word cloud. Right: Visualization options for the word cloud.

- Select the number of words in the word cloud (minimum is 3).

Visualization options:

- Choose the word cloud theme. Dark mode is the default one. There are other themes such as Barbie, Jungle, Sunset, and Default(the default theme of wordcloud2 function in wordcloud2 library)
- Select the shape of the word cloud. Options are circle(default), cardioid, diamond, triangle-forward, triangle, pentagon and star.
- Select size of the word cloud from 1 to 2.5 (default is 1.2). When you increase the size, if the words do not fit the page they disappear.
- Remove specific words in the word cloud (maximum 10 specific words).
- Select the number of words in the word cloud (minimum is 3).
- Report a bug or see the code with the given link.

In the Fig.15 the word cloud shows the words of genre 'Religious' with the theme is 'Barbie' between 1984-2017. As you can see, the number of songs is 52. The word 'lord' is used 14 times.

3.7.3 On the main panel:

- The word cloud of the chosen genre is shown.
- When user pass over the words with mouse, the number of words that is used in the lyrics is shown at the left-down corner of the rectangle.
- There is an explanation of the word cloud in the below of the word cloud image. The chosen publication years and number of songs are displayed.

There are two other browsers in the app. "What is world cloud?" explains why word cloud is used in some cases. Also, there is an explanation of stopwords with examples. "About this app" gives information about the Data Visualization course and some explanation of how to use the app. The Fig.16 shows the other tabs.


```
[1] "Number of songs between the years 1984 and 2017 is 52"
```

Figure 15: The word cloud representation of the lyrics of genre 'Religious' with Barbie theme. The number of songs and publication years are shown in below.

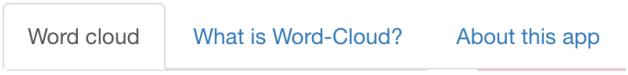

Figure 16

References

[1] Sankey diagram. Page Version ID: 1062562883. URL: https://en.wikipedia.org/w/index.php?title=Sankey_diagram&oldid=1062562883.

[2] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[3] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3: data-driven documents. 17(12).

[4] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*, pages 336–343. ISSN: 1049-2615. doi:10.1109/VL.1996.545307.