An exploration of music lyrics over time

FIT5147 - Data Visualisation Project

Jonathan Skorik - 25978675

Music lyrics have changed quite a bit over time. In this report, we detail how we have created an application using the Shiny package in R that aids the user in investigating how different words have changed in popularity over time. Through the use of word clouds, the user is able to see how words have generally changed over time. This is coupled with scatter plots to show the user exactly how these words have changed over time. The main focus of the application is to show users how swear words have changed over time.

1 Introduction.

In the exploration component of this project, we aimed to investigate trends in music lyrics over time. To do this, we used the lyrics from the top 100 songs according to Billboard Magazine [1] from 1965 to 2015. The conclusions we found included:

- The average number of words in songs has been increasing.
- The amount of unique words in songs has been decreasing, suggesting more repetitive lyrics in songs.
- Certain words have followed interesting trends over time. For example, the word 'love' has not changed much in popularity while the word 'baby' has been increasing.
- Swear words were not popular in songs until the early 1990s and since then have grown to be relatively popular.

Of these findings, the most interesting is how swear words did not start to become popular until the 1990s. This will form the basis for the narrative we want to depict here. Our aim here is to provide visualisation that gives user an idea of how words have changed in popularity over time while putting emphasis on the trend for swear words over time. The target audience for this visualisation is music enthusiasts.

The aim of this report to describe the design process for a visualisation focused around displaying this information and providing a framework for people to investigate their own words of interest. The remained of this report will be structured in the following way: Section 2 will discuss the design process followed to obtain our final visualisation. In Section 3, we will then discuss how we went about implementing our idea and the problems that occurred during implementation. Section 4 will consist of a user guide explaining how the visualisation can be used. Finally, in Section 5 we will conclude our work by reflecting upon the work done and suggest improvements.

2 Design

In this section, we will explore the design process that lead us to our final design and implementation. To design this visualisation, the Five Design Sheet methodology was implemented. In Appendix A, we provide the five design sheets for this visualisation which we created using the aid of templates found here [2].

In the first design sheet we discuss what findings were already investigated in the exploration component of this project. We also discussed further findings that may aid in our final design such as investigating TF-IDF weightings for the words used in song lyrics over time. Of the findings discussed, the most interesting was that swear words did not start to become popularised in music until the early 1990s. This became the main focus of what we wanted to present and convey. Accompanying this, it was decided that we also wanted to present word clouds over time to give a broad overview of how words changed in popularity over time. Overall, the structure of our visualisation is to have word clouds that initially take the users attention and gives them a broad overview of how words have changed in popularity over time. Following this, the user will then be directed to focus on how the popularity of swear words changed over time. This changes their focus to a more narrow subset of words opposed to all possible words shown in the word clouds.

Design sheets 2, 3, and 4 were designed with the idea of starting with a base design full of static visualisations then push the possibilities of the user interactions with each following design sheet.

In design sheet 2, we present a very simple design whereby the user is presented with word clouds showing how the popularity of words has changed over the decades (1960s to 2010s). Following this, they are able to select different words to view how many songs they have been used in over time. The main bulk of these words were to be swear words but would also contain some general words that were

found to be interesting in the exploration component. This design is very simple and light in user interactivity but forms the basis for possibilities explored in the remaining design sheets.

In design sheet 3, we increased the user interactivity available for the word clouds. This design allowed for the year of the word cloud to be chosen by the user or be set to increment in year every few seconds. This allows the user to get an idea of how words change more accurately from year to year opposed to just over each decade. For this design, the user interaction available for the word popularity over time plots did not change.

In design sheet 4, we heavily pushed the amount of interactivity for the visualisation. We first increased the breadth of the word clouds by deciding to present three opposed to one. The word clouds now presented show the number of times a word is used in each year, the number of songs a word appears in, and the Term Frequency - Inverse Document Frequency (TF-IDF) weighting for each word in each year. It can be noted that the TF-IDF weightings for words was not calculated in the previous component of this project and so it was done specifically for this implementation. It was calculated by combining all songs for a year into a single 'document' which was then compared to all other years combined songs to find the TF-IDF values for each word in each year. Furthermore, for this design, we also increased the amount of user interactivity available to investigate words over time. Instead of having a set number of words a user can investigate, we now allow the user to type in any word they want and investigate it. The initial words shown to the user will be swear words. This is to emphasise the narrative we want to depict.

The main positives of this design is that the user gets a very broad overview of how words have changed over time but then they are able to analyse specific words of their choosing. The two visualisations are intended to work together - the word clouds give the user ideas of words to investigate while the scatter plots give them the means to focus their investigation on a subset of words of their choosing. The main concern for this design is that is may be slow to run. Each time the year is changed for the word clouds, three new word cloud figures need to be generated. This could be circumvented by having the word clouds for each year generated when the application is started and only recall them each time the year changes, but this will lead to a slow start up time. Alongside this, generating the word clouds for any word chosen by the user can also take time. Although this design gives the user a large amount of interactivity, it is at the potential cost of a slower running application.

In design sheet 5, we present our final design for our visualisation. This visualisation is a slightly modified version of design sheet 4 where we have included text alongside our plots to aid in explaining how to use the visualisation and also push our narrative more. The design includes text that suggests to the user to change the year to investigate the word clouds over time. We also provide text to explicitly tell the user that swear words were not popular until the early 1990s and that we have provided scatter plots to show this. Text is then used to lead the user into investigating their own words of choice to find out how they have changed in popularity over time.

Up until this point, we have only discussed the layout of the application and not finer details of each type of visualisation explicitly. The word clouds will use a combination of colour and size to depict the importance of words. The word clouds are structured such that the most important words are in the center and as the words move away from the center, they become less important. The words that are in the center of the word cloud are yellow and as the words get further from the center they start becoming orange and then eventually red. The words that most important are also larger than the less important words and so the words get smaller the further away from the center they are. Following this, it is clear that larger words are yellow and smaller words are red. This is important as if we had the colours the other way around, it could be quite difficult to read the smaller words. Having the smaller words be the darker colour makes them much more readable while having the larger words be the lighter colour does not take away from their readability. This colour scale is effective as it allows for easy differentiation between the importance of words when coupled with size.

For the scatter plots, we have decided to assign each word its own colour simply to keep them differentiated from each other. We have also made the concious decision to facet the scatter plots so that each word has its own axis as if we did not, the plot could become quite busy. If we allowed all

words to be plotted on the same axis, then just over 50 data points would be added to the axis for each word. Even for a small amount of words, this can be difficult to interpret.

3 Implementation.

The first step to implement this visualisation was to wrangle the initial data set found on Kaggle [3]. This data set had to be wrangled into three new data sets containing information for word frequencies and TF-IDF values. The final form of all three data sets was tabular.

Due to the addition of the feature allowing users to choose any words they want to plot, the idea of generating all required plots on initialisation is no longer practical. As a result, we need to be able to generate plots dynamically. Due to this constraint, we decided to develop this application in R as a Shiny application as it is likely we will be required to frequently filter data sets. As our data sets are in a tabular form, this operation is quite simple in R and thus using it will simplify the implementation. Opposed to writing the entire Shiny application in a single file, we opted to write it over two files (server file and user interaction file) as this is a requirement to publish it online.

The R libraries we utilised to create this visualisation were:

- shiny: Required to create a Shiny Application.
- ggplot: Used to generate the scatter plots showing the percentage of songs containing given words over time.
- ggwordcloud: Used to create the word cloud plots.
- readr: Required to read in the data sets, both locally or externally.
- colourspace: Used to select the colours for the word clouds.

On top of utilising R code, we also decided to utilise basic HTML to format our application nicely. Using HTML, we also added extra features to our application that were not initially present in our final design sheet. This included links to the initial data set and the wrangled data sets and also tooltips.

As aforementioned, we made the decision to generate all plots to be presented when required. The main purpose for this is because it would be very time and space consuming to generate and store the scatter plots for every word on initialisation. This increased the complexity for the implementation as we now needed to find a method to generate the plots based on user input. To overcome this, functions where the input was the input given by the user and the output was the required plot were utilised. In the case of the slider for years and the word clouds, if the slider value changed, functions would be called that generated the three new word clouds based on the year selected by the slider. For the scatter plot, when the words entered in the text box changed, they would be passed to a function that would generate and return a new ggplot object containing the plots for the specified words. The plots returned by these functions would replace the plots displayed to the user.

4 User Guide.

In this section, we provide a user guide for how to operate the visualisation designed and implemented. There are two possible methods to run the application implemented in this work. First, it can simply be ran locally through RStudio using the files submitted. To do this, extract the files from the submission folder, open either of the 'ui.R' or 'server.R' files in RStudio and press the 'Run App' button. Once running, increase the size of the page to be full screen to correctly show the visualisation.

The second method is to run the application externally. The application that was designed and implemented here was published to `shinyapps.io`. The application can be found here: `https://jtskorik.shinyapps.io/MusicLyricsOverTime/`. The data sets required were also uploaded to a

public GitHub repository here: `https://github.com/JTSkorik/MusicLyricData`. The web version of the application obtains the required data sets from the GitHub repository. The submitted version can also use the data sets on GitHub opposed to the provided data sets but this would require an internet connection to run the application.

Please note that in either case, if the data is being obtained from GitHub, it will take an extra few seconds to start up the application.

Now we have discussed how to start the application, we will provide a detailed guide of the different operations that can be done within the application. In Figure 1, we have provided a snapshot of the application and have highlighted components in boxes. In the green box, we highlight the slider that allows the user to choose what year they want to view word clouds for. The blue box shows the button the user can press to have the slider increment in years every few seconds. The titles highlighted by the purple boxes contain tooltips that explain the weights of the word clouds more thoroughly. For example, if a mouse was to hover of the title 'Number of Songs Containing', a tooltip would appear stating 'Ranked based on how many songs contain each word in each year'.

In Figures 2 and 3, we further examine the application by looking by the scatter plot component. In both figures, the red box contains a text box that allows for the user to enter words. As we can see by looking at Figure 2 then Figure 3, if the user changes the words in the text box, the words plotted in the scatter plots also change. We can again note that the application starts by displaying the plots for swear words as this forms the main component of the narrative we want to show.

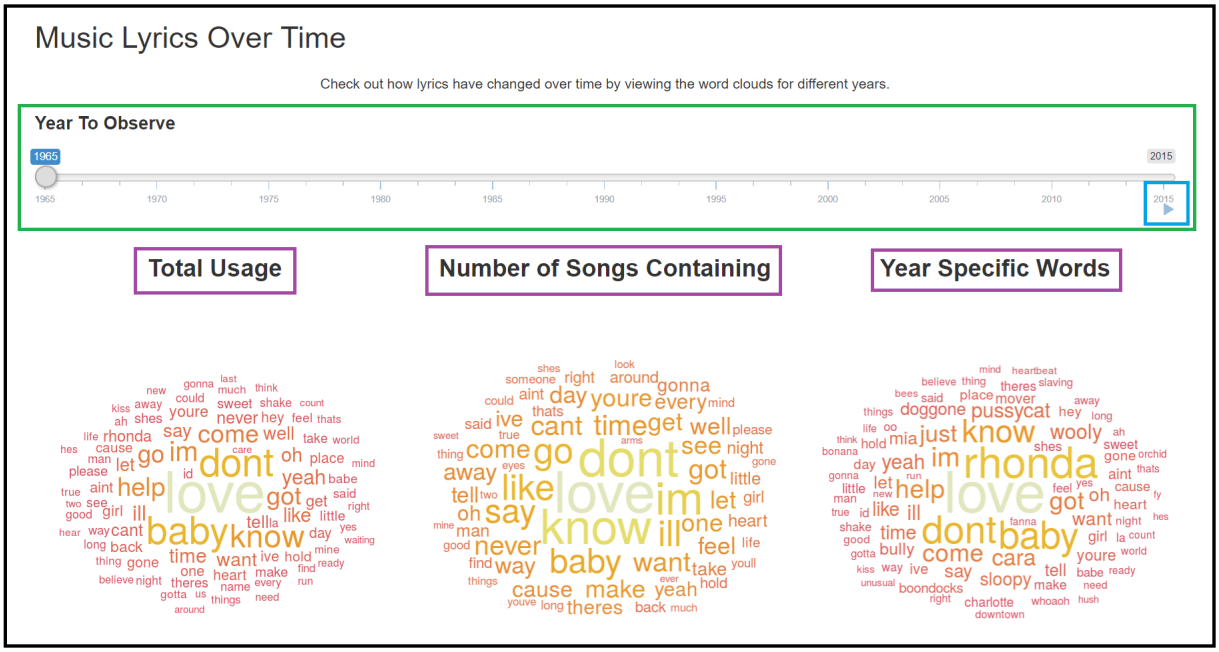


Figure 1: Sign posting for the possible user interactions for the word clouds using the green, blue, and purple boxes.

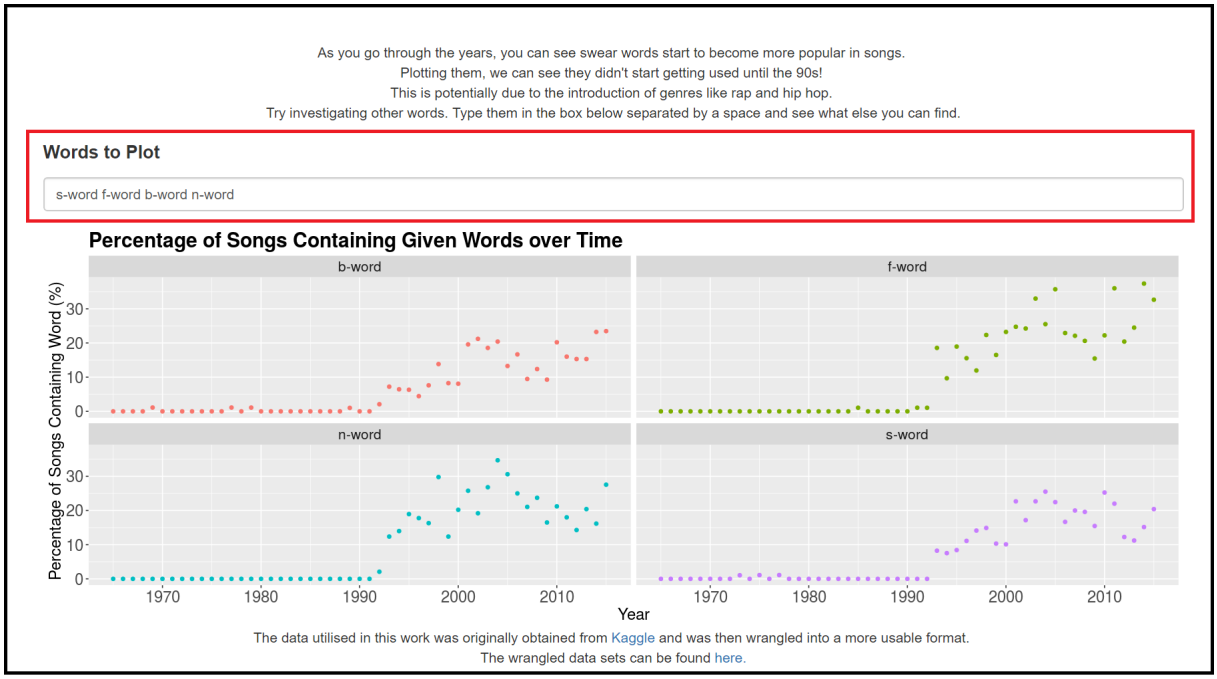


Figure 2: Sign posting for the possible user interactions for the scatter plots that show the percentage of songs containing each word over time.

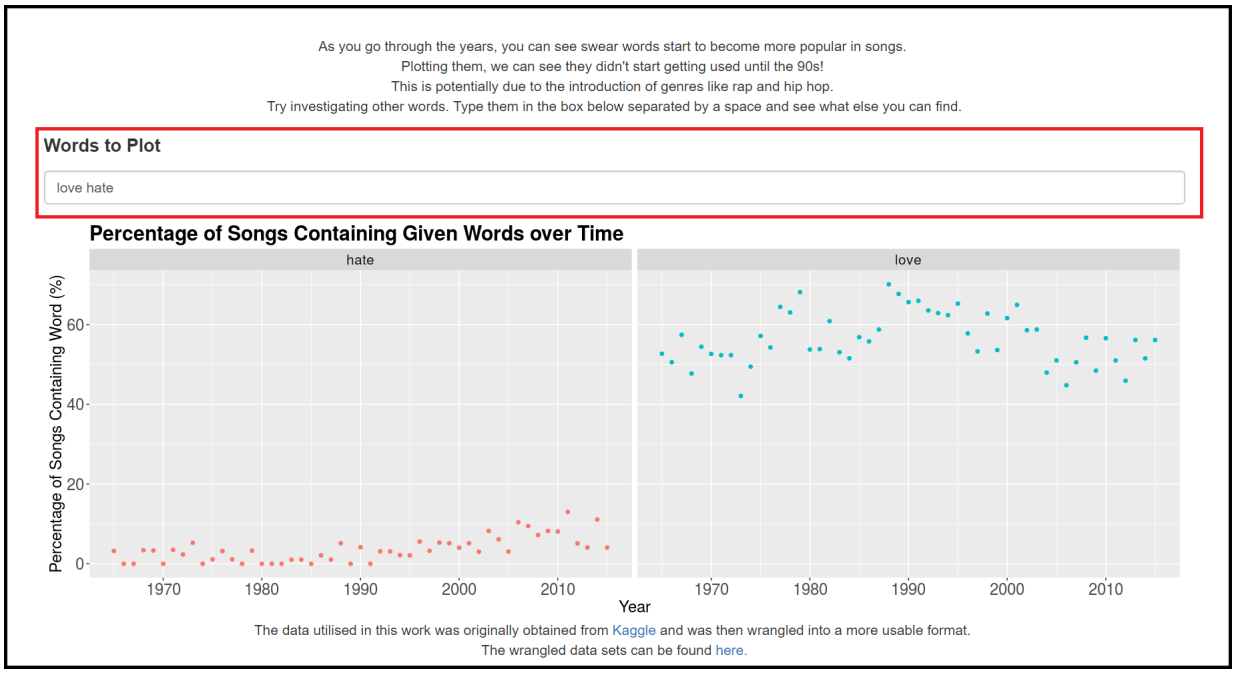


Figure 3: Example to show how when the words are changed in the text box, the words plotted also change.

5 Conclusion.

In this work, we discuss the implementation of an interactive visualisation usable by music enthusiasts to analyse how the popularity of lyrics has changed over time. We created an application which allows

users to get a broad idea of how words have changed in popularity over time through the use of word clouds. The application then allows the user to investigate any words of their choosing to observe what percentage of songs they occurred in over time. The main narrative for this visualisation was to show users how swear words have increased in popularity in music. We push this narrative by explicitly showing this to the user but also provide the ability for the user to investigate any word of their choosing.

To further extend the work here, we may consider adding an extra level of interactivity to the scatter plots that allow the user to determine what songs contribute to data points. The idea being that a user could move their mouse over a particular point in the scatter plot and a tooltip would appear telling the user the songs and artists that contribute to that data point. This will allow the users to investigate points to find what is contributing to them. For example, the s-word is used in Pink Floyd's 'Money' in 1973 but most of the occurrences of the s-word took place after the early 1990s, making this point an outlier. The addition of this aspect would allow the user to easily discover what song caused this outlier. To do this, we would consider transferring to D3/JavaScript due to the mouse over interactivity. We would also have to consider the issue that if a word is in a lot of songs for a given year, there would be a lot of information to display. This could cause an information overload.

References

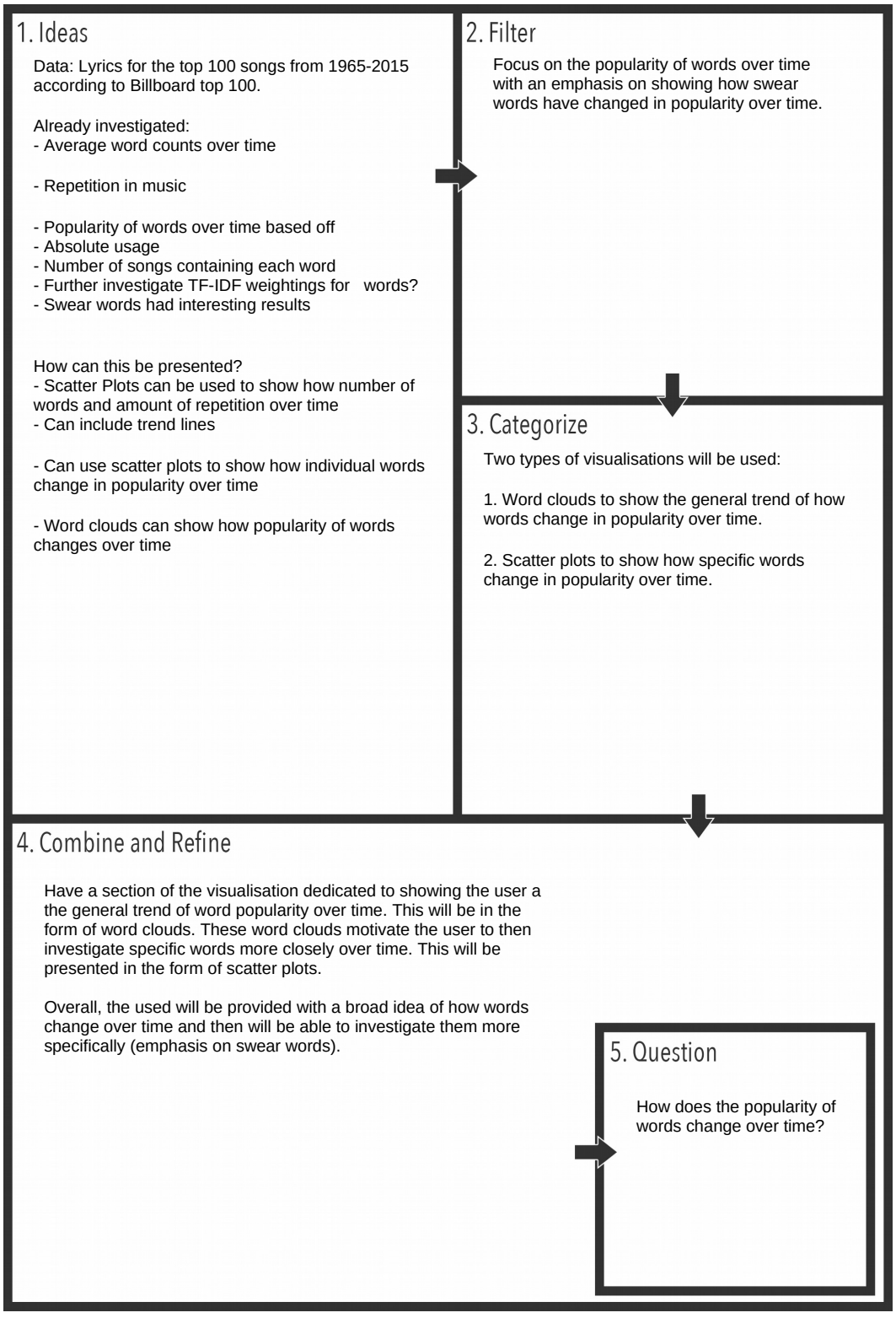
[1] http://billboardtop100of.com, 2019.

[2] http://fds.design/, 2019.

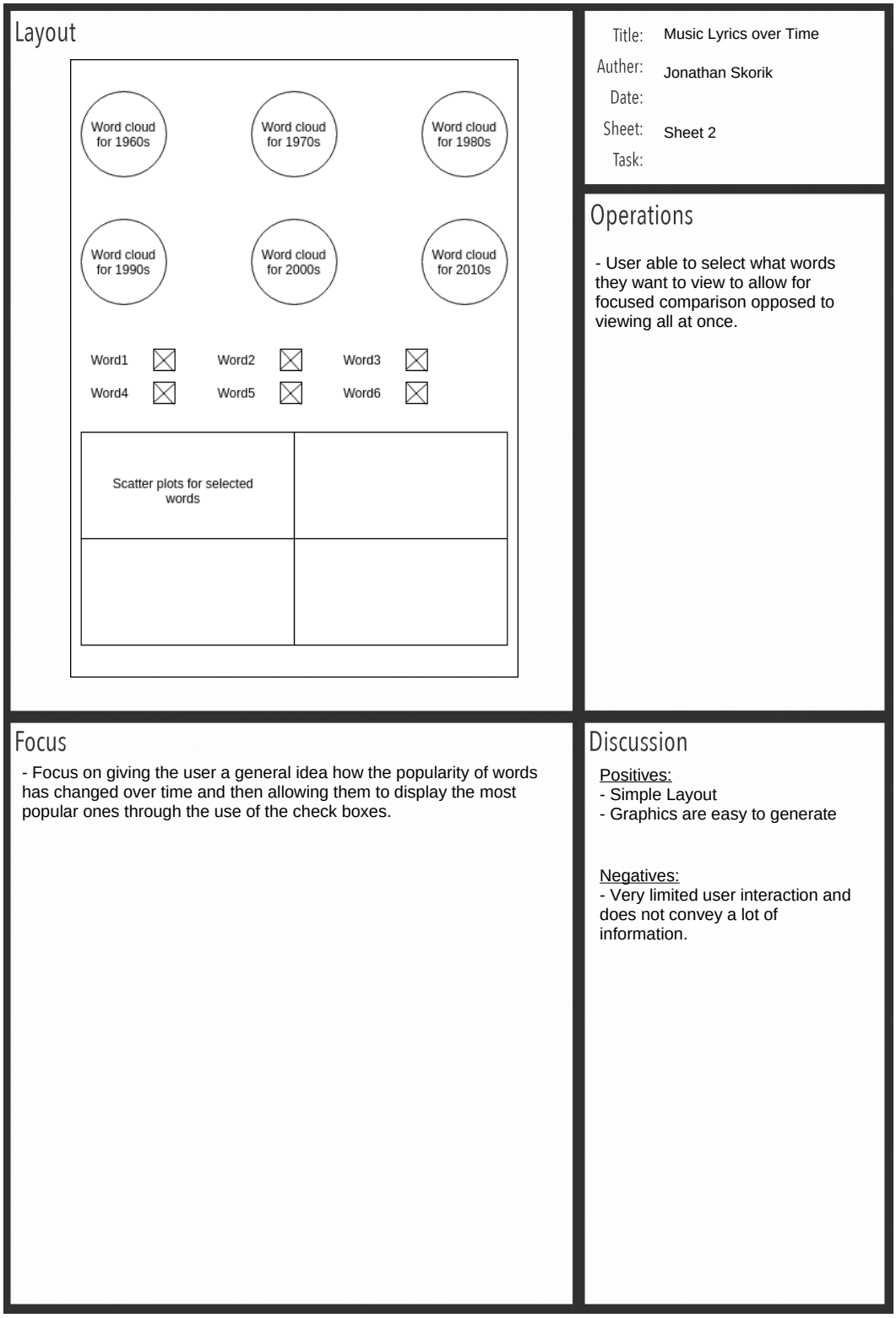
[3] https://www.kaggle.com/rakannimer/billboard-lyrics, 2019.

Appendix A.

Design Sheet 1.

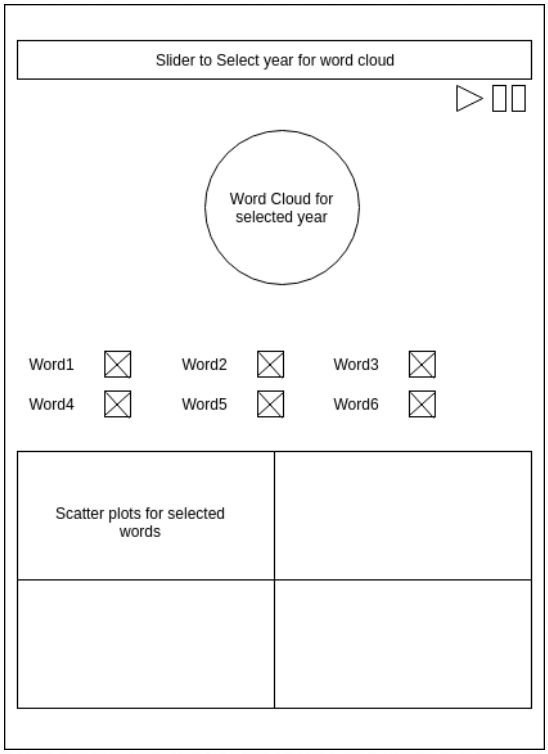


Design Sheet 2.



Design Sheet 3.

Layout



Title: Music Lyrics over Time
Auther: Jonathan Skorik
Date:
Sheet: Sheet 3
Task:

Operations

- User is able to decide what year they want to show from the word cloud.

- User able to set the word cloud to automatically iterate through years.

- User able to select what words to show in the scatter plots from a selection of popular words.

Focus

- Allowing users to view the word clouds specific to each year to give a more focused overview of the popularity of words over time.

- Allows user to more specifically focus on certain words that are available.

- General transition from a broad over view of words over time from the word clouds to a more specific view in the scatter plots.

Discussion

Positives:
- Refined information given by word clouds (based on each year opposed to decade).

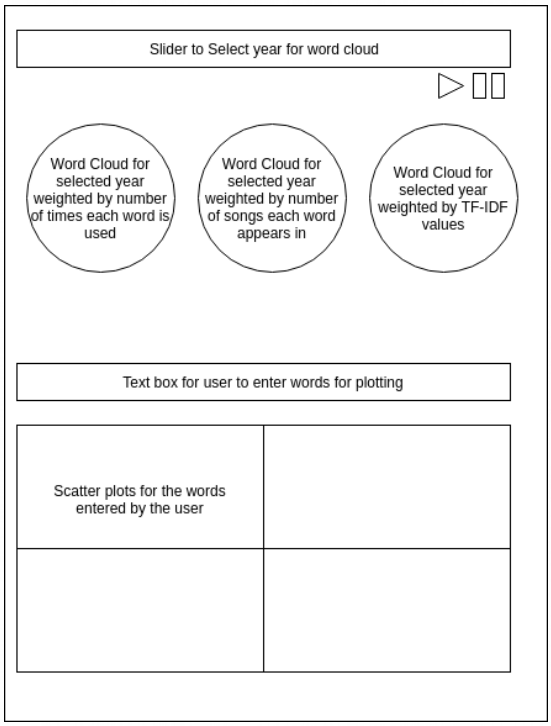
- Simple design

Negatives
- Word clouds may be to slow to generate on the fly.

- User is limited by the number of words they can observe thoroughly at once.

Design Sheet 4.

Layout



Title: Music Lyrics over Time
Auther: Jonathan Skorik
Date:
Sheet: Sheet 4
Task:

Operations

- Slider to select the year for the word clouds

- Play/Pause button that allows the year to be incremented every few seconds

- User can input any words they like to compare and see how they trend over time

Focus

- Strong focus on given the user a broad over view of interesting words to investigate through word clouds weighted by the number of songs a word appears in, the number of times the word appears in general, and the popularity of words in each year compared to other years (TF-IDF).

- Full interactivity with the scatter plots. User can choose any word at all that they might want to investigate.

- Visualisation starts with broad view of words over time and is then the user can focus on any words they want to.

- Focus on having scatter plot initially showing swear words.

Discussion

Positives:
- User can plot any word of their choice, not limited to a select few.

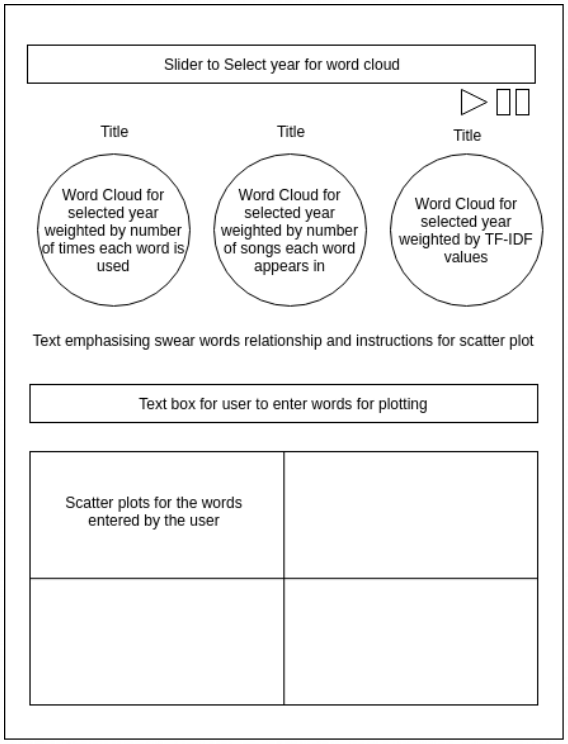
- Three types of word clouds to give more general ideas for the user to investigate.

Negatives:
- May be slow to generate each of the word clouds and scatter plots on the fly.

-Questionable whether the interactive features can actually be implemented.

Design Sheet 5.

Layout



Title: Music Lyrics over Time
Auther: Jonathan Skorik
Date:
Sheet: Sheet 5
Task:

Operations

- Slider to select the year for word clouds

- Play/Pause button that incrementally increases the year every few seconds for word clouds

- User is able to input any word they want to analyse further

Focus

- Focus on providing broad and narrow ways for a user to investigate the change in popularity of words over time.
- Word clouds allow for a broad overview of how words have changed in popularity while the scatter plots allow for specific word analysis.
- Visually, the word clouds appear first followed by the scatter plots so the user goes from broad to narrow focus.

- Start with the scatter plot component initially displaying the change in swear words over time as this is the main point to be shown.

- Large focus on user interactivity. Want users to be able to explore the data set graphically themselves.

Detail

- Will be completed using a shiny application in R.

- Easy to generate each individual visualisation using wordcloud package and ggplot.

- Need to be able to generate plots based on a text input.

- Can be computationally intensive to generate these plots on the fly, causing a laggy application. (Maybe produce and store all word clouds when application starts opposed to generating on the fly?)