

From Objects to Data, Final Project

Bruno Sotic

11353473

Helge Moes

11348801

Bruno Koldewej

12289434

Work Group 30

Dr. ir. J. Kamps

Final Project: <https://grinfandango.github.io/FO2D.CORRECT/>

Data and Raw Data:

https://drive.google.com/drive/folders/1YkVrYJW0yEcE9ZU2Wf9BwHPnf4b_n30q?usp=sharing

Repository and Code: <https://github.com/GrinFandango/FO2D.CORRECT>

Introduction

This document is a documentation and reflection on the progress and learning outcome that were made for the course From Objects to Data. The project itself consists of an empirical study, paying respect to the field of popular music studies, and is grounded in a digital humanities approach. Our research study is built on recent academic outputs within the domain of musicology and cognition. More precisely, our output provides additional argumentations of a novel musical genre that many researchers have classified as ‘‘situational music’’. Whilst research in this field is continuously studying what makes people like certain music and what role music has in the life of individuals, contemporary studies have continuously shown that individuals tend to choose, favour and use a specific type of music for specific contexts or tasks. However, this is still understudied which also makes it an interesting phenomenon to scrutinize upon.

The study (i.e. Research report) is presented in an academic format on our projects page:

<https://grinfandango.github.io/FO2D.CORRECT/>

Therefore, this document will not pay close attention to the theoretical research aspect. Rather, this paper is a reflection of the technical process behind it. This means that it emphasises the programmatic and methodological details.

The goals of this project were formulated as the following:

- Engage in a quantitative research project
- Get familiar with research topic, concepts, methods and theories within the domain of digital music studies
- Get familiar with programming in R and RStudio (since we have some knowledge of Python, this would provide us with additional space to reflect upon computational methods)
- Learn to work with the Spotify API
- Work with online surveys and learn how to use survey instruments that are appropriate for the study
- Perform a demographic analysis
- Get a better understanding of digital objects (in this case, songs) and of the information that is used to construct them (audio features and qualities)
- Work with GitHub
- Present the research in a data-journalistic manner in the form of a website

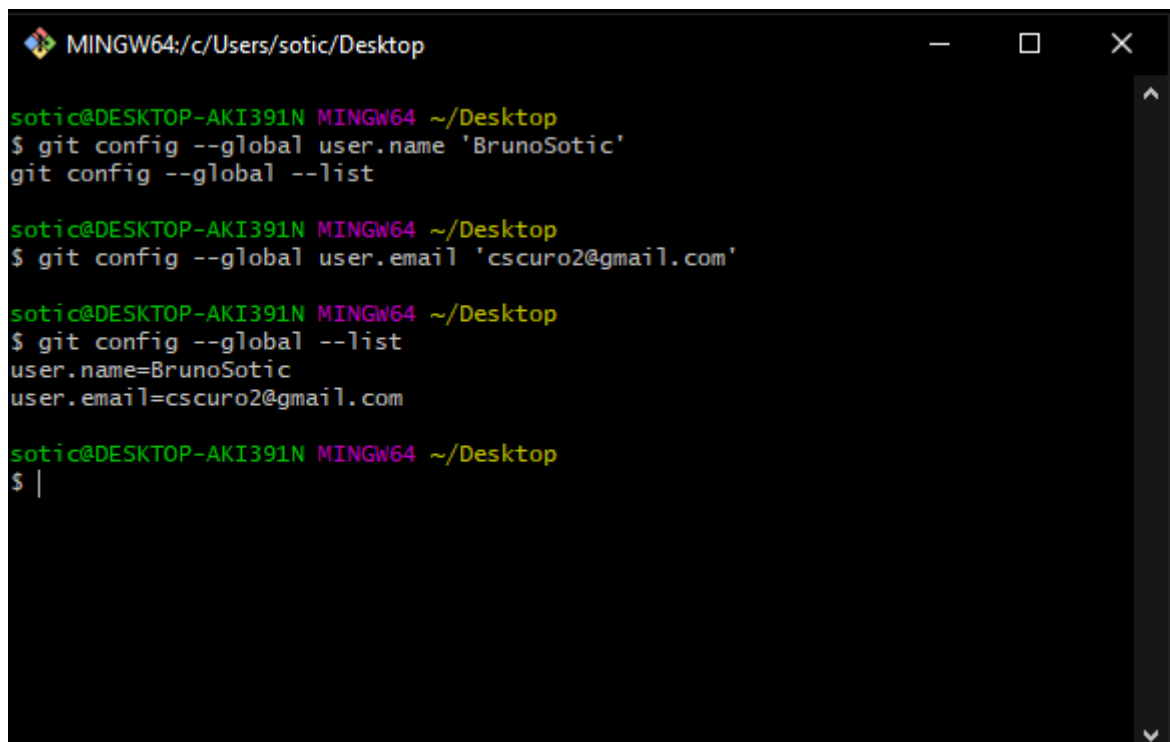
Additionally, it is important to note that it was impossible for us document every step of the way. We made use of numerous online tutorials and examples. We started out with basic free courses on DataCamp (Introduction to Tydiverse, visualization with Ggplot and Communicating with data).

Additionally, guides and walkthroughs were very useful that were found on GitHub, Reddit, YouTube, Medium.com and towardsdatascience.com

GitHub

Following the GitHub documentation *HappyGit* from chapter 4 to 15, we initiated the repository with a *ReadMe* file and enabled the github.pages feature. This is then used to host our project and present our final output (research report) as a website / webpage.

Git and Gitbash

A screenshot of a Windows terminal window titled "MINGW64:/c/Users/sotic/Desktop". The terminal shows the following commands and output:

```
sotic@DESKTOP-AKI391N MINGW64 ~/Desktop
$ git config --global user.name 'BrunoSotic'
git config --global --list

sotic@DESKTOP-AKI391N MINGW64 ~/Desktop
$ git config --global user.email 'cscuro2@gmail.com'

sotic@DESKTOP-AKI391N MINGW64 ~/Desktop
$ git config --global --list
user.name=BrunoSotic
user.email=cscuro2@gmail.com

sotic@DESKTOP-AKI391N MINGW64 ~/Desktop
$ |
```

Since GitHub is used to store and host our project as a repository, Git is used to establish a connection between the machine and the account associated with the repository. This was at first alien to us, especially since we were not too familiar with terminals and shells. While all of us managed to connect, we were facing some troubles with connecting and making changes to one single repository. We eventually enabled and used an SSH URL to clone the repository to our devices, rather than the standard HTTPS. We could now all make changes to the repository.

The repo. is then opened in Rstudio:

File -> New Project -> Version Control -> Git -> Repository URL

Spotify API

Spotify's Application Programming Interface is a set of functions that allows researchers and application developers to access the features and data from its platform / service. Spotify is currently one of the leading music streaming platforms. However, what makes the platform so attractive to research is the diverse variety it offers users to access information about objects (in our case, songs). Since it primarily hosts audio objects, the API enables one to retrieve information about the qualities that together make up the objects (e.g. A wide range of audio features, melodic patterns etc). In addition, one is also able to retrieve data associated to the objects (such as: artists and their relationship to other artists, most popular songs over given time frames...). This digital method approach offers tremendous potential for research in the humanities domain. For example, we are now able to access and visualize information that shows us how musical genres have evolved over time; or we can assess how the qualities of popular music have changed over the course of the past 10 years. While the API provides exciting modes to inquire into music from various perspectives, it does offer some limitations due to its discreteness. For example, we found it impossible to find an explanation of how the system calculates or establishes how what the values of a song are, in terms of its audio features.

Connecting to Spotify

Install.packages('spotifyr')

```
1 Sys.setenv(SPOTIFY_CLIENT_ID = '4ad8e6ae5a9e45efacbc2cf00b82b2dc')
2 Sys.setenv(SPOTIFY_CLIENT_SECRET = '0a173a2dcd41415b8266fcb27fecb6fe')
3
4 access_token <- get_spotify_access_token()
```

Spotify API: Access and Authorization

As explained on Spotify's API page, one first needs to create a developer account for the platform. The user would then be given a client ID and a "client secret". As we understood it, this basically acts as an identifier to the system to establish who is getting access to the application system, but also as a means to protect other users on the platform. Someone with a developer account cannot obtain any information of another user or of the music the particular user listens to. This is only possible if the other users give permission to the developer, which is confirmed through the IDs and client secrets. Finally, the user needs to obtain authentication and permission to access the API. This step is done through the function *get_spotify_access_token()*. This is visible on line 4 in the figure above. Since the authorization function is tedious to write up, it is stored under the variable *access_token* by using the *<-* function in R.

R and Rstudio: Exploration Examples

While Spotify provides a detailed and elaborative explanation on the use of the API, translating all of it to R was at first trivial. Therefore, we followed tutorials and examples from YouTube,

Medium.com, TowardsDataScience.com and GitHub. This would eventually emphasize the role and importance of indexing on the internet, since we would be met with a lot of issues, problems and question marks; but also interesting and feasible results. While we did not document every experiment and the whole learning process (since it would result in too much of the same material), the following are some illustrations of our earlier learning outcomes.

```
1 #FO2D: Test Spotify API
2
3 library(spotifyr)
4
5 Sys.setenv(SPOTIFY_CLIENT_ID = '4ad8e6ae5a9e45efacbc2cf00b82b2dc')
6 Sys.setenv(SPOTIFY_CLIENT_SECRET = '0a173a2dcd41415b8266fcb27fecb6fe')
7
8 get_spotify_access_token()
9
10 library(tidyverse)
11 library(knitr)
12
13 FO2D.test <- get_artist_audio_features('the beatles')
14 FO2D.test %>%
15   count(key_mode, sort = TRUE) %>%
16   head(5) %>%
17   kable()
18 |
19
```

Spotify API: Exploration: Beatles Major Key Frequency: Code

The code above was a good exploration of getting familiar with how the API works. After a connection and permission was obtained from Spotify (lines 1 – 8), we used a basic function to retrieve audio information regarding the band the Beatles. This data can be explored in endless ways. In this example, we were using the packages that we had worked with during our learning phase; namely tidyverse and knitr. Those packages are also commonly used ones and present in almost every tutorial we encountered so far. In short, the packages can be described as sets of pre-defined functions, mainly used for the transformation, visualization, modelling and communication of data.

The `get_artist_audio_features()` function on line 13 is self explanatory. It retrieves information on audio features for tracks (songs) that are associated to an artist based on the artists name on Spotify. The artists name is specified with quotation marks and enclosed with brackets. The function that returns the information for the artist (in this case the Beatles) is then stored in a variable value that we name `FO2D.test`, by using `<-`. Here, we understood that this is also stored as a data table (or data frame) in the *environment interface* in Rstudio, which means we can see the raw data in the form of a column and row based spreadsheet. The function would in this case return 139 entries and 39 columns, showing audio information for the Beatles. This is then passed on to another function by using `%>%`. `Count()` is used

```
Console Terminal x Jobs x
~/FO2D.Group30/ ↵
> FO2D.test <- get_artist_audio_features('the beatles')
> FO2D.test %>%
+   count(key_mode, sort = TRUE) %>%
+   head(5) %>%
+   kable()

|key_mode|  n|
|:-----|:--|
|D major| 24|
|G major| 21|
|A major| 13|
|F major| 12|
|C major| 11|
```

Execution & Result

From the data returned we can then conclude that the favourite chord key of the Beatles is in D major, while they used C major the least.

This is then stored as a data frame under the defined value *FO2D.test*; which is saved, visible and accessible in the 'Environment' window in R studio, meaning it can be repurposed for later use.

```
← → Source on Save
1 #FO2D: Test Spotify API
2
3 library(spotifyr)
4
5 Sys.setenv(SPOTIFY_CLIENT_ID = '4ad8e6ae5a9e45efacbc2cf00b82b2dc')
6 Sys.setenv(SPOTIFY_CLIENT_SECRET = '0a173a2dcd41415b8266fcb27fecb6fe')
7
8 get_spotify_access_token()
9
10 library(tidyverse)
11 library(knitr)
12
13 FO2D.testX <- get_track_audio_features('25yQPHgC35WnnnOUqFhgVR', get_spotify_access_token())
14
```

Code: Scrape: Beatles Song: Audio Features

The code above is a basic version of what our final information retrieval process would look like. The function returns audio features for the song (track) that is specified within the brackets. The song must be specified with its track ID on Spotify. For the final dataset, the function would look the same, but contain multiple song IDs (separated by a comma). The screenshot below presents what the output looks like.

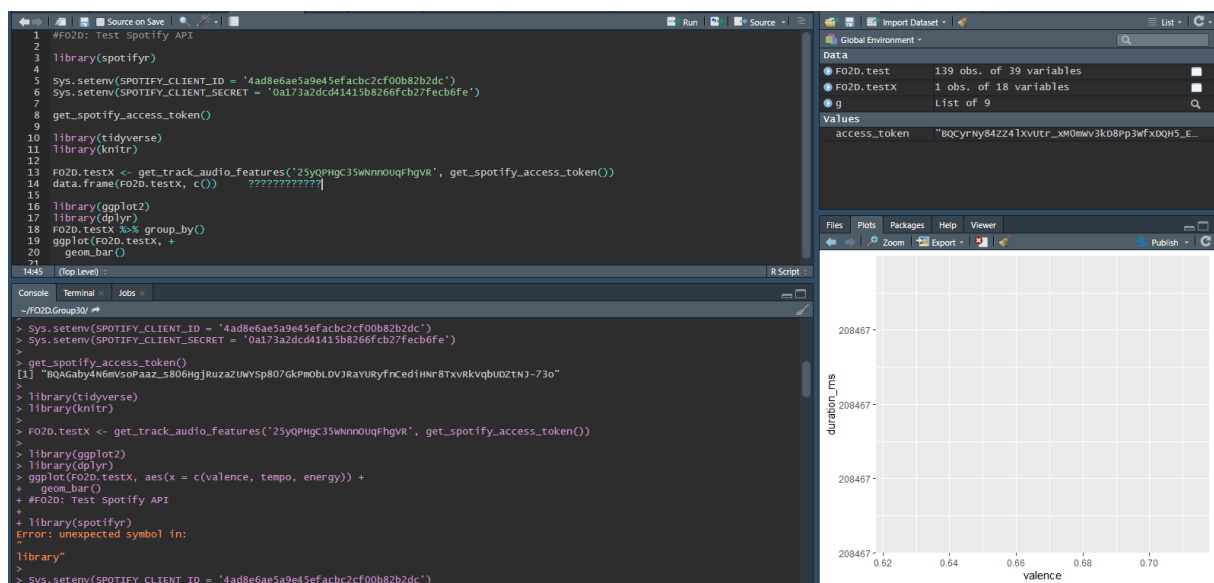
```
~/FO2D.Group30/ ↗
>
> get_spotify_access_token()
[1] "BQC3H0S3x4-C0NyPPoIDN-QHEryi60Wwe_M-Cf8Z5VHXIkR8M-m1kDL_-1HQ6A7oM1aN_zzbmCHih13Sarw"
>
> library(tidyverse)
> library(knitr)
>
> FO2D.testX <- get_track_audio_features('25yQPHGc35WnnnOuqFhgvr', get_spotify_access_token())
> FO2D.testX
# A tibble: 1 x 18
  danceability energy key loudness mode speechiness acousticness instrumentalness liveness valence tempo type id uri track_href
  <dbl> <dbl> <int> <dbl> <int> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr> <chr> <chr> <chr>
1 0.311 0.325 2 -9.04 1 0.0283 0.0469 0 0.139 0.668 65.1 audi~ 25yQ~ spot~ https://a~
# ... with 3 more variables: analysis_url <chr>, duration_ms <int>, time_signature <int>
>
```

R Execution

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	type
1	0.311	0.325	2	-9.042	1	0.0283	0.0469	0	0.139	0.668	65.09	audio_features

Result: Data Table

However, here we faced a lot of difficulties with wrangling the dataset. We will reflect on this issue in detail on a later stage in this document (when we discuss working with our survey data). The code bellow illustrates a first failed attempt to draw meaning out of the table and visualization.



After this, we returned back to DataCamp to rehearse the chapter on Tidyverse and Communication with Ggplot. Afterwards, we also sourced out more tutorials for working with the Spotify API (mainly various blogposts and github). After some innitial learning and different experimentations, we were able to work with the data retrieved from spoitfy.

```
library(spotifyr)
library(tidyverse)
library(knitr)
library(ggjoy)
library(ggplot2)

Sys.setenv(SPOTIFY_CLIENT_ID = '4ad8e6ae5a9e45efacbc2cf00b82b2dc')
Sys.setenv(SPOTIFY_CLIENT_SECRET = '0a173a2dcd41415b8266fcb27fecb6fe')

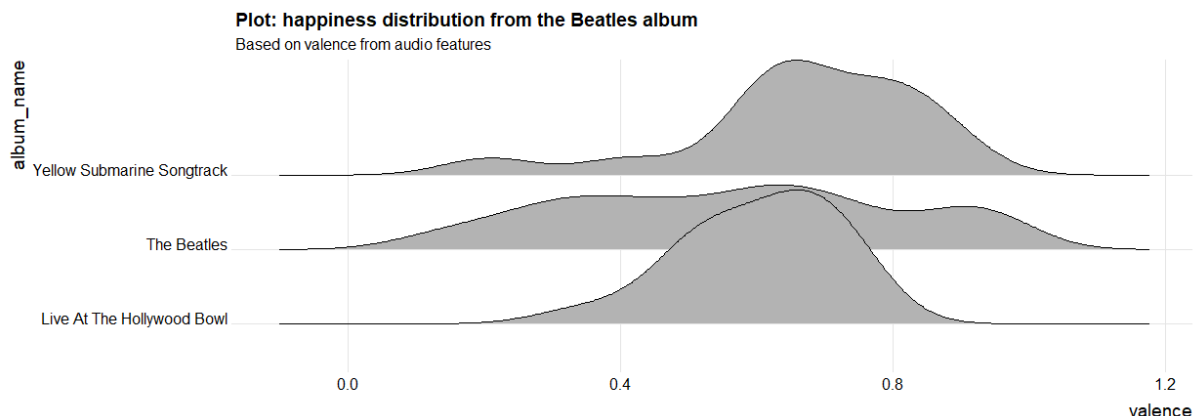
access <- get_spotify_access_token()

BeatlesF <- get_artist_audio_features('the beatles')

BeatlesF %>%
  arrange(-valence) %>%
  select(track_name, valence) %>%
  head(5) %>%
  kable()

ggplot(BeatlesF, aes(x = valence, y = album_name)) +
  geom_joy() +
  theme_joy() +
  ggtitle('Plot: happiness distribution from the Beatles album', subtitle = "Based on valence from audio features")]
```

In this example, we explored how positive certain songs are from the Beatles and how this perceptual ‘‘melodic positiveness’’ is distributed within different albums. Once the audio features for every song from the artist are retrieved, this dataset is arranged based on the column valence. This is then generated as another table using the *knitr()* function. The data is visualized using *ggplot* and pays respect to the album where the valence falls under (song in the album). Finally, *geom_joy* is a function which acts almost as an extension of *ggplot*, that orders the data as presented below.



Survey (Data collection, datasets and instruments)

As a primary mean for data collection, an online survey was constructed through Google Forms. The details of, and instruments used for the survey will not be addressed in great detail in this paper, since it is explained on our projects website portfolio. In short, we firstly asked participants how often they listen to music while studying (either passively or actively). The answer could be indicated on a scale from ‘‘never’’ to ‘‘always’’. If the answer was ‘‘never’’ the survey would end. This is because the role of music while studying is not the focus of our paper, but rather the musical qualities and dynamics of that music.

Finally, we gathered information about basic demographics, such as: age, gender, faculty etc., and about their personalities. The later was done by using the Ten Item Personality Inventory (TIPI), where the participants would rank some of their personality traits on a 7 point item scale. This is a widely used and cited method for demographic research. The weight of their answers would then be factorized and their final score would indicate which of the 5 big personalities they belong to. The datasets would eventually be filtered based on those demographics, together with the songs they provided. The figures below illustrate a snippet of what our final dataset contained.

	A	B	C	D	E	F	G	H	I
1	We would like to ask if you are currently studying	Do you ever listen to music	#1 Song Study	#2 Song Study	#3 Song Study	#1 Song Anti-Study	#2 Song Anti-Study	#3 Song Anti-Study	
2	I understand the text pres No								
3	I understand the text pres Yes	Rarely		https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/
4	I understand the text pres Yes	Never		https://youtu.be/WPnI755-	-	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/
5	I understand the text pres Yes	Sometimes		https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/
6	I understand the text pres Yes	Often		https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/
7	I understand the text pres Yes	Rarely		https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/
8	I understand the text pres Yes	Rarely		https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/
9	I understand the text pres Yes	Often		spotify:track:32sVB5BHr spotify:track:4XJ06VtHFKF spotify:track:5BgqPUSAV spotify:track:7KQJtSC28s spotify:track:763PMQZ2s spotify:track:50r7Kt07Kq	spotify:track:1PxM5MAK3 spotify:track:3Qa9480TMtM spotify:track:2Yv2mHzr5A spotify:track:23fgrC18SFs spotify:track:6c1e1QWQZz spotify:track:41rWj13Hvq	spotify:track:32sVB5BHr spotify:track:4XJ06VtHFKF spotify:track:5BgqPUSAV spotify:track:7KQJtSC28s spotify:track:763PMQZ2s spotify:track:50r7Kt07Kq	spotify:track:1PxM5MAK3 spotify:track:3Qa9480TMtM spotify:track:2Yv2mHzr5A spotify:track:23fgrC18SFs spotify:track:6c1e1QWQZz spotify:track:41rWj13Hvq	spotify:track:32sVB5BHr spotify:track:4XJ06VtHFKF spotify:track:5BgqPUSAV spotify:track:7KQJtSC28s spotify:track:763PMQZ2s spotify:track:50r7Kt07Kq	spotify:track:1PxM5MAK3 spotify:track:3Qa9480TMtM spotify:track:2Yv2mHzr5A spotify:track:23fgrC18SFs spotify:track:6c1e1QWQZz spotify:track:41rWj13Hvq
10	I understand the text pres Yes	Sometimes		https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/
11	I understand the text pres Yes	Sometimes		https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/
12	I understand the text pres Yes	Rarely		https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/
13	I understand the text pres Yes	Always		https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/	https://open.spotify.com/https://open.spotify.com/
14	I understand the text pres Yes	Often		spotify:track:QpqrWJf5W spotify:track:0dbqPssU7k spotify:track:0uYiq080XE spotify:track:0nQJMT8wz spotify:track:7IAa7WJ11S spotify:track:2wLw0ZQ8B	spotify:track:3YYKm3IGC spotify:track:31Q55w6G0 spotify:track:58e9wAPO spotify:track:1B3ix5C2zT spotify:track:6dKl0xk7Eh spotify:track:70uIdZ2Ue	spotify:track:QpqrWJf5W spotify:track:0dbqPssU7k spotify:track:0uYiq080XE spotify:track:0nQJMT8wz spotify:track:7IAa7WJ11S spotify:track:2wLw0ZQ8B	spotify:track:3YYKm3IGC spotify:track:31Q55w6G0 spotify:track:58e9wAPO spotify:track:1B3ix5C2zT spotify:track:6dKl0xk7Eh spotify:track:70uIdZ2Ue	spotify:track:QpqrWJf5W spotify:track:0dbqPssU7k spotify:track:0uYiq080XE spotify:track:0nQJMT8wz spotify:track:7IAa7WJ11S spotify:track:2wLw0ZQ8B	spotify:track:3YYKm3IGC spotify:track:31Q55w6G0 spotify:track:58e9wAPO spotify:track:1B3ix5C2zT spotify:track:6dKl0xk7Eh spotify:track:70uIdZ2Ue

Extraverted, enthusiastic	Critical, quarrelsome	Dependable, self-disciplined	Anxious, easily upset	Open to new experiences	Reserved, quiet	Sympathetic, warm	Disorganized, careless	Calm, emotionally stable	Conventional, uncreative
3	5	6	4	4	6	5	4	3	1
ack/2Fxmhs0bxGSBdJ92M42m									
6	5	4	2	6	2	6	2	6	4
3	6	7	6	6	3	6	3	3	1
5	5	5	2	6	5	6	2	5	3
5	5	5	3	5	3	5	2	5	3
5	6	6	3	6	2	6	1	6	3
3	3	5	4	5	3	6	5	3	4
7	1	5	3	4	2	6	1	6	3
6	4	6	2	6	1	5	4	5	3
6	4	4	5	5	2	6	6	3	2
7	6	6	2	6	3	6	3	6	2
6	5	6	5	6	6	6	2	3	

What is your age (in years)?	What is your gender?	What is your field of study?	What is your nationality?
23	Male	Economics	Dutch
22	Male	Mechanical engineering	Dutch
21	Female	Data Science	Dutch
21	Male	Economics & Business	Dutch
24	Male	Philosophy	Dutch
23	Male	Engineering	Dutch
23	Male	International relations	Dutch
21	Female	Communication	Dutch
20	Female	Business Administration	Dutch
21	Female	Graphic Design	Dutch
21	Female	Philosophy/sociology/politics	Dutch
21	Female	Creative Business	(you ain't much if you ain't Dutch)

Snippet: Raw Data: Survey Results

Accessible at:

https://drive.google.com/drive/folders/1YkVrYJW0yEcE9ZU2Wf9BwHPnf4b_n30q?usp=sharing

Here, we were also met with one of the most time consuming and tedious issue, that is, importing (and working with) an external excel or .csv file in Rstudio. We encountered this problem before while we were still getting familiar with R. We are not sure why, but whatever option or method we tried, we could not wrangle or access an excel or .csv file through Rstudio. While we went back to the tutorial on Data Camp, it became apparent that we have only worked with data sets that are constructed within R so far. We were looking for the errors and troubleshooting options online (YouTube, blogs, Reddit, Github, Stackoverflow) and did not resolve the issue. We then thought that we might be missing some essential packages, but that was also not the case. We installed the library(readxl) and followed numerous tutorials step-by-step, but we still did not make progress. We thought that Rstudio might not know what path to take to find file (even though it is in the folder with all the code and visible in the environment interface). However, specifying that as recommended by other users online did also not work.

```
data <- read_excel('C:\Users\sotic\Documents\FO2D.CORRECT\AgeAndGender.xlsx')
```

Setting our directory with *stewd()* and specifying the path to the document was also unsuccessful. We downloaded and our survey results numerous times in different formats, but that still didn't work. We did realize that the way in which the tables are constructed in RStudio and via Google Sheets have fundamental differences (i.e. One uses dots and the other commas for decimal values), but formatting them back and forth was also not the solution. Later, we would come to understand that the file needs to be specified in R, meaning that R needs to receive instructions on how to read the columns and rows. At the end we also realized that all the data sets we were working so far were in a vertical structure, whereas ours was horizontal. While we did make a lot of progress with that while working on our project, that was mainly at the very end of our project already. We also realized that re-constructing the data set in R using the *data.frame()* function would make it work. But given that we have a very large dataset (169 songs and a wide range of demographic information), inputting that as a *data.frame()* would be very time consuming. Therefore, a fair part of the calculations, manipulation and wrangling was done in MS Excel. We would then import the results we needed from the calculations in R. This might not have been the most ideal option, but it was the most efficient way to troubleshoot around the problem. In addition, we also realized that gaining insight from those data sets required statistical experience, for which we are uncertain if we would have managed to do properly in R (given that quantitative methods are absent at our study program).

What is essential for retrieving information about audio objects through the API is to have a songs (tracks) unique Spotify ID. This is present in the Spotify URI. We firstly filtered the dataset from

figure one into 2 separate files, containing the IDs for ‘study songs’ and ‘anti-study songs’. We then imported the IDs into RStudio manually. The information for each of the songs is retrieved using:

```
get_track_audio_features('ID1, ID2, ID3...', get_spotify_authorization_token())
```

This is done for both study and anti-study music. A fundamental limitation was that Spotify only allows one to retrieve information for 100 songs maximum at once. Therefore, we had to pull information for each of the 2 data sets 3 times, respectively. Finally, adding and dividing those values. We also exported the data, a snippet from the raw dataset is illustrated bellow.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
		danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	type	id	uri	track_href	analysis_url	duration_ms	time_signature	mention_male_students	mention_male_athletes	mention_female_students	mention_female_athletes
1	1	0.73	0.867	11	-5.881	1	0.032	0.0395	0.00E+00	0.0861	0.776	104.019	audio_feature_3d0d44CM	spotify:track:https://api.spotify.com/v1/tracks/3d0d44CM	https://api.spotify.com/v1/tracks/3d0d44CM	https://api.spotify.com/v1/tracks/3d0d44CM	https://api.spotify.com/v1/tracks/3d0d44CM	200373	4	0	0	1	0
2	2	0.589	0.196	8	-17.281	0	0.0454	0.979	1.03E-02	0.221	0.362	114.616	audio_feature_6zIMAsFg	spotify:track:https://api.spotify.com/v1/tracks/6zIMAsFg	https://api.spotify.com/v1/tracks/6zIMAsFg	https://api.spotify.com/v1/tracks/6zIMAsFg	https://api.spotify.com/v1/tracks/6zIMAsFg	192160	4	1	0	0	0
3	6	0.7540	0.801	1	-5.992	0	0.35	0.184	0.00E+00	0.132	0.391	90.843	audio_feature_3ipz2xH1	spotify:track:https://api.spotify.com/v1/tracks/3ipz2xH1	https://api.spotify.com/v1/tracks/3ipz2xH1	https://api.spotify.com/v1/tracks/3ipz2xH1	https://api.spotify.com/v1/tracks/3ipz2xH1	208867	4	1	0	0	0
4	7	0.3400	0.489	7	-15.316	1	0.0412	0.83	9.10E-01	0.141	0.468	157.024	audio_feature_10je92vyl	spotify:track:https://api.spotify.com/v1/tracks/10je92vyl	https://api.spotify.com/v1/tracks/10je92vyl	https://api.spotify.com/v1/tracks/10je92vyl	https://api.spotify.com/v1/tracks/10je92vyl	167880	3	1	0	0	0
5	9	0.8170	0.599	0	-9.249	0	0.0328	0.132	3.11E-04	0.0873	0.548	108.873	audio_feature_2KH15Wvi	spotify:track:https://api.spotify.com/v1/tracks/2KH15Wvi	https://api.spotify.com/v1/tracks/2KH15Wvi	https://api.spotify.com/v1/tracks/2KH15Wvi	https://api.spotify.com/v1/tracks/2KH15Wvi	245640	4	0	0	0	0
6	10	0.7230	0.752	7	-5.146	1	0.143	0.0804	0.00E+00	0.102	0.423	108.064	audio_feature_1PKm5MA	spotify:track:https://api.spotify.com/v1/tracks/1PKm5MA	https://api.spotify.com/v1/tracks/1PKm5MA	https://api.spotify.com/v1/tracks/1PKm5MA	https://api.spotify.com/v1/tracks/1PKm5MA	191103	4	1	0	0	0
7	11	0.5670	0.267	4	-6.502	1	0.0299	0.839	1.45E-06	0.089	0.0592	110.011	audio_feature_12MCOx7	spotify:track:https://api.spotify.com/v1/tracks/12MCOx7	https://api.spotify.com/v1/tracks/12MCOx7	https://api.spotify.com/v1/tracks/12MCOx7	https://api.spotify.com/v1/tracks/12MCOx7	240133	4	0	0	1	0
8	12	0.5590	0.23	7	-20.178	0	0.0309	0.805	8.76E-01	0.0776	0.461	151.987	audio_feature_54QF6IOI	spotify:track:https://api.spotify.com/v1/tracks/54QF6IOI	https://api.spotify.com/v1/tracks/54QF6IOI	https://api.spotify.com/v1/tracks/54QF6IOI	https://api.spotify.com/v1/tracks/54QF6IOI	289280	4	0	0	1	0
9	13	0.9310	0.517	4	-8.476	1	0.105	0.0923	8.21E-01	0.114	0.171	118.021	audio_feature_0UvqXlyC	spotify:track:https://api.spotify.com/v1/tracks/0UvqXlyC	https://api.spotify.com/v1/tracks/0UvqXlyC	https://api.spotify.com/v1/tracks/0UvqXlyC	https://api.spotify.com/v1/tracks/0UvqXlyC	168814	4	0	0	1	0
10	14	0.4890	0.404	8	-7.815	1	0.0261	0.487	4.85E-02	0.14	0.153	108.010	audio_feature_0qprwJfj	spotify:track:https://api.spotify.com/v1/tracks/0qprwJfj	https://api.spotify.com/v1/tracks/0qprwJfj	https://api.spotify.com/v1/tracks/0qprwJfj	https://api.spotify.com/v1/tracks/0qprwJfj	597600	4	0	0	1	0
11	15	0.4840	0.192	2	-15.378	1	0.0465	0.991	9.08E-01	0.106	0.0559	100.084	audio_feature_3YKrm3Kc	spotify:track:https://api.spotify.com/v1/tracks/3YKrm3Kc	https://api.spotify.com/v1/tracks/3YKrm3Kc	https://api.spotify.com/v1/tracks/3YKrm3Kc	https://api.spotify.com/v1/tracks/3YKrm3Kc	140733	4	0	0	1	0
12	17	0.7020	0.615	8	-8.541	1	0.0286	0.777	1.62E-05	0.173	0.579	116.982	audio_feature_5iGrd1UA	spotify:track:https://api.spotify.com/v1/tracks/5iGrd1UA	https://api.spotify.com/v1/tracks/5iGrd1UA	https://api.spotify.com/v1/tracks/5iGrd1UA	https://api.spotify.com/v1/tracks/5iGrd1UA	140478	4	0	0	1	0
13	18	0.7910	0.329	10	-12.906	0	0.0592	0.6	8.63E-01	0.193	0.526	127.021	audio_feature_4i8HMeDb	spotify:track:https://api.spotify.com/v1/tracks/4i8HMeDb	https://api.spotify.com/v1/tracks/4i8HMeDb	https://api.spotify.com/v1/tracks/4i8HMeDb	https://api.spotify.com/v1/tracks/4i8HMeDb	230500	4	0	0	1	0
14	19	0.2280	0.639	7	-6.457	0	0.0635	0.411	7.22E-06	0.299	0.23	187.931	audio_feature_51Grd1UA	spotify:track:https://api.spotify.com/v1/tracks/51Grd1UA	https://api.spotify.com/v1/tracks/51Grd1UA	https://api.spotify.com/v1/tracks/51Grd1UA	https://api.spotify.com/v1/tracks/51Grd1UA	215933	4	0	0	1	0
15	20	0.4420	0.00919	3	-32.452	1	0.047	0.984	9.36E-01	0.0873	0.0929	79.314	audio_feature_1VnvvEs	spotify:track:https://api.spotify.com/v1/tracks/1VnvvEs	https://api.spotify.com/v1/tracks/1VnvvEs	https://api.spotify.com/v1/tracks/1VnvvEs	https://api.spotify.com/v1/tracks/1VnvvEs	273667	4	0	0	1	0
16	22	0.7320	0.41	9	-14.402	1	0.0814	0.87	8.86E-01	0.109	0.658	82.980	audio_feature_1p9bEdnL	spotify:track:https://api.spotify.com/v1/tracks/1p9bEdnL	https://api.spotify.com/v1/tracks/1p9bEdnL	https://api.spotify.com/v1/tracks/1p9bEdnL	https://api.spotify.com/v1/tracks/1p9bEdnL	121446	4	0	0	1	0
17	23	0.5070	0.837	8	-4.955	0	0.359	0.0849	0.00E+00	0.232	0.754	80.562	audio_feature_2qrULwL1	spotify:track:https://api.spotify.com/v1/tracks/2qrULwL1	https://api.spotify.com/v1/tracks/2qrULwL1	https://api.spotify.com/v1/tracks/2qrULwL1	https://api.spotify.com/v1/tracks/2qrULwL1	311040	4	1	0	0	0
18	29	0.5320	0.204	2	-19.805	1	0.0477	0.851	9.60E-01	0.108	0.0399	120.090	audio_feature_6S8ysPC6	spotify:track:https://api.spotify.com/v1/tracks/6S8ysPC6	https://api.spotify.com/v1/tracks/6S8ysPC6	https://api.spotify.com/v1/tracks/6S8ysPC6	https://api.spotify.com/v1/tracks/6S8ysPC6	105627	3	1	0	0	0
19	30	0.6850	0.737	8	-8.465	1	0.0395	0.17	1.56E-03	0.166	0.247	123.997	audio_feature_1SXZBDJ	spotify:track:https://api.spotify.com/v1/tracks/1SXZBDJ	https://api.spotify.com/v1/tracks/1SXZBDJ	https://api.spotify.com/v1/tracks/1SXZBDJ	https://api.spotify.com/v1/tracks/1SXZBDJ	147853	4	0	0	1	0
20	33	0.6900	0.644	11	-7.392	0	0.376	0.657	1.12E-05	0.112	0.818	78.995	audio_feature_4vyrDnOC	spotify:track:https://api.spotify.com/v1/tracks/4vyrDnOC	https://api.spotify.com/v1/tracks/4vyrDnOC	https://api.spotify.com/v1/tracks/4vyrDnOC	https://api.spotify.com/v1/tracks/4vyrDnOC	255094	4	0	0	1	0
21	35	0.1040	0.0948	9	-20.890	0	0.0359	0.876	9.58E-01	0.101	0.0344	78.515	audio_feature_7E99x7JIC	spotify:track:https://api.spotify.com/v1/tracks/7E99x7JIC	https://api.spotify.com/v1/tracks/7E99x7JIC	https://api.spotify.com/v1/tracks/7E99x7JIC	https://api.spotify.com/v1/tracks/7E99x7JIC	135707	4	0	0	1	0
22	36	0.8280	0.182	1	-13.066	0	0.166	0.754	7.06E-02	0.15	0.231	147.279	audio_feature_1kwnxJNv	spotify:track:https://api.spotify.com/v1/tracks/1kwnxJNv	https://api.spotify.com/v1/tracks/1kwnxJNv	https://api.spotify.com/v1/tracks/1kwnxJNv	https://api.spotify.com/v1/tracks/1kwnxJNv	330507	4	0	0	1	0
23	32	0.2550	0.0253	0	-25.806	1	0.0415	0.993	9.55E-01	0.0947	0.649	87.037	audio_feature_75ZDycp7	spotify:track:https://api.spotify.com/v1/tracks/75ZDycp7	https://api.spotify.com/v1/tracks/75ZDycp7	https://api.spotify.com/v1/tracks/75ZDycp7	https://api.spotify.com/v1/tracks/75ZDycp7	116792	4	0	0	0	0
24	37	0.3500	0.0109	0	-26.688	1	0.0392	0.99	9.10E-01	0.109	0.203	131.369	audio_feature_6NiE7nOFr	spotify:track:https://api.spotify.com/v1/tracks/6NiE7nOFr	https://api.spotify.com/v1/tracks/6NiE7nOFr	https://api.spotify.com/v1/tracks/6NiE7nOFr	https://api.spotify.com/v1/tracks/6NiE7nOFr	130332	1	0	0	1	0
25	38	0.7230	0.422	6	-7.276	0	0.125	0.153	7.43E-01	0.177	0.54	94.044	audio_feature_6ghMpqd	spotify:track:https://api.spotify.com/v1/tracks/6ghMpqd	https://api.spotify.com/v1/tracks/6ghMpqd	https://api.spotify.com/v1/tracks/6ghMpqd	https://api.spotify.com/v1/tracks/6ghMpqd	152553	4	1	0	0	0
26	40	0.9390	0.305	8	-10.952	0	0.354	0.35	1.69E-04	0.105	0.563	100.029	audio_feature_3Tc57i9i2	spotify:track:https://api.spotify.com/v1/tracks/3Tc57i9i2	https://api.spotify.com/v1/tracks/3Tc57i9i2	https://api.spotify.com/v1/tracks/3Tc57i9i2	https://api.spotify.com/v1/tracks/3Tc57i9i2	179889	4	0	0	1	0
27	41	0.4480	0.406	2	-11.420	1	0.0483	0.791	2.31E-03	0.112	0.0504	147.912	audio_feature_3K844ssjX	spotify:track:https://api.spotify.com/v1/tracks/3K844ssjX	https://api.spotify.com/v1/tracks/3K844ssjX	https://api.spotify.com/v1/tracks/3K844ssjX	https://api.spotify.com/v1/tracks/3K844ssjX	210640	4	0	0	1	0

Audio Features: Raw Data

Accessible at:

https://drive.google.com/drive/folders/1YkVrYJW0yEcE9ZU2Wf9BwHPnf4b_n30q?usp=sharing

This information could now be computed and made ready for our analysis. We firstly filtered the dataset to exclude unnecessary information (such as: url, type, duration etc.) so that only the values relation to audio qualities are left. We also excluded ‘key’ from the dataset, since the numbers for this feature are used as indicators for which chord is used the most in a song. Finally, a *mean()* and *median()* were calculated for each of the features. We then proceeded to filter our survey dataset per demographic. For example:

Age	Count										
19	6										
20	11										
21	26										
22	10										
23	4										
24	5										
26	1										
27	1										
Average Age	21,32813										
Gender	Count										
Male	26										
Female	37										
Other/Prefer not to say	3										
Male Study	danceabilit	energy	key	loudness	mode	speechine	acousticne	instrument	liveness	valence	tempo
Average	0,625764	0,492141	5,126437	-12285	0,574713	0,088154	0,434338	0,371962	0,164847	0,378516	114209,839
Male Antistudy	danceabilit	energy	key	loudness	mode	speechine	acousticne	instrument	liveness	valence	tempo
Average	0,623575	0,730534	4,505747	-6655,94	0,482759	0,144693	0,177193	0,095745	0,201247	0,51442	119775,172
Female Study	danceabilit	energy	key	loudness	mode	speechine	acousticne	instrument	liveness	valence	tempo
Average	0,558528	0,414239	5,642276	-13451,2	0,634146	0,081085	0,542263	0,462804	0,13895	0,388577	120355,22
Female Antisudy	danceabilit	energy	key	loudness	mode	speechine	acousticne	instrument	liveness	valence	tempo
Average	0,671372	0,678547	5,61157	-6904,03	0,553719	0,124997	0,227782	0,047311	0,207725	0,581693	119745,479

Survey Statistics: Male & Female Demographic

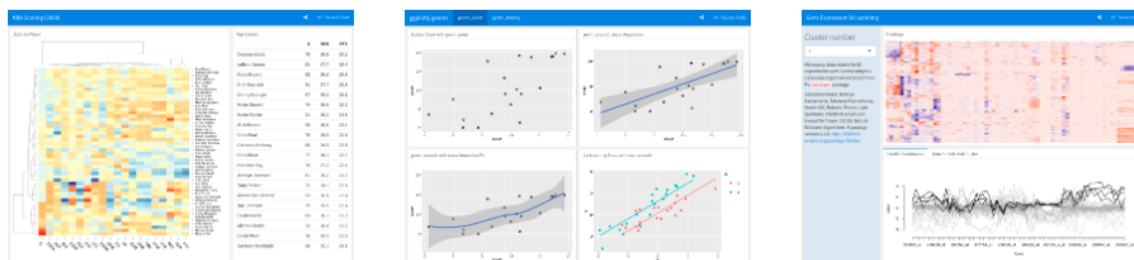
Accessible at:

FlexDashboard

In order to push out and present our research in a data-journalistic manner, we initially made use of github.pages and one of its custom templates. This would give a scrollable webpage to present our results. However, while doing research on outputs as such, we came across the *flexdashboard* package. Using standard html markdown elements, this package made it easy to construct an interactive environment to present our outputs.

flexdashboard: Easy interactive dashboards for R

- Use [R Markdown](#) to publish a group of related data visualizations as a dashboard.
- Support for a wide variety of components including [htmlwidgets](#); base, lattice, and grid graphics; tabular data; gauges and value boxes; and text annotations.
- Flexible and easy to specify row and column-based [layouts](#). Components are intelligently re-sized to fill the browser and adapted for display on mobile devices.
- [Storyboard](#) layouts for presenting sequences of visualizations and related commentary.
- Optionally use [Shiny](#) to drive visualizations dynamically.



Getting Started

Install the **flexdashboard** package from CRAN as follows:

```
install.packages("flexdashboard")
```

To author a flexdashboard you create an [R Markdown](#) document with the `flexdashboard::flex_dashboard` output format. You can do this from within RStudio using the **New R Markdown** dialog:

Since it written in a Rmarkdown format, we quickly understood how to work with it. We were already somewhat familiar with how text is represented through html, so we found this experience relatable to writing a report in a LaTeX program, such as Overleaf. Except that we could add chinks of R code to it, which specified as:

```
```{r}

Code

```
```

After a sketch was made of how we believe the output should be presented (according to academic outputs of design principles), we imported the flexdashboard template and created a structure.

File -> New File -> R Markdown -> From Template -> FlexDashboard

The following figure illustrated the standard structure that comes with the template (the code will be explained later on):

```

1 ---
2 title: "Untitled"
3 output:
4   flexdashboard::flex_dashboard:
5     orientation: columns
6     vertical_layout: fill
7 ---
8
9 ```{r setup, include=FALSE}
10 library(flexdashboard)
11 ```
12
13 column {data-width=650}
14 -----
15
16 ### Chart A
17
18 ```{r}
19
20 ```
21
22 column {data-width=350}
23 -----
24
25 ### Chart B
26
27 ```{r}
28
29 ```
30
31 ### Chart C
32
33 ```{r}
34
35 ```

```

Console

```

~/R02DGroup30/
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'Citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>

```

Flexdashboard provides an elaborative and easy-to-follow guide on how to structure the dashboard, using functions and elements that belong to the package. The code on lines 1 till 7 would define what type of document it is, in which format to output it (usually html, but in this case `flex_dashboard`), and some basic details. The *orientation* argument would establish how the data should be outputted. This can be either *columns* or *rows*. Here we also found some significant limitations as well as issues, which will be elaborated further.

Column output example:

```

1 ---
2 title: "Column Orientation"
3 output: flexdashboard::flex_dashboard
4 ---
5
6 Column
7 -----
8
9 ### Chart 1
10
11 ```{r}
12
13 ```
14
15 Column
16 -----
17
18 ### Chart 2
19
20 ```{r}
21
22 ```
23
24 ### Chart 3
25
26 ```{r}
27
28 ```

```



Row output example:

```
1 |---
2 |title: "Chart Stack (Scrolling)"
3 |output:
4 |  flexdashboard::flex_dashboard:
5 |    vertical_layout: scroll
6 |  ---
7 |
8 |  ### Chart 1
9 |
10 |  {{{{r}}}
11 |  {{{{r}}}
12 |
13 |  ### Chart 2
14 |
15 |  {{{{r}}}
16 |  {{{{r}}}
17 |
18 |  ### Chart 3
19 |
20 |  {{{{r}}}
21 |  {{{{r}}}
22 |
23 |
24 |
25 |
```



Here we faced some trials and error since the layout is very limited by the dashboard's operational logic. E.g. it is impossible to have one chart be in the form of a column and the other in the form of a row. The *row* and *column* argument are also essential for determining how the data in it will be compressed and presented. While Rows are good for wide graphs (such as bar charts), columns are better for more central graphs. This would eventually be a problem since some information and its visualization does not play well with the pre-defined data orientation. It is possible to tweak the sizes and positions of the charts, but the way in which the data will be presented in it is still subject to the orientation argument.

Ultimately, the way we got accustomed to it is by trying every possible mode. The output would oftentimes be different than what we would expect. Some formats would not even render the data that's inside; or the charts would just be all over the place.

The layout is defined by using a level 2 markdown header. In our case:

```
555
556 column
557 -----
```

This would establish the functionality for a column. The column is the established and named by using:

Name of column

If one wants another column that would appear to the left or right of the previous column, then another column argument needs to be typed below. However, if one wants a column to be positioned underneath the previous column, then he or she needs to add a second column name only (### Name column 2). This is a simple formula, but it took us quite some time to figure out.

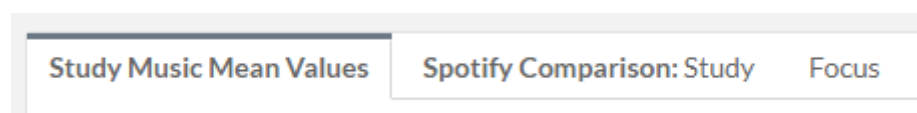
Additionally, one is able to specify the sized (height and width) of the column. After some initial research, we understood that the size of a website is commonly understood as being 1000 “big”. This means that a value of 500 would take up half of the screen for instance. The sizes are defined like the following:

```
142  
143 column {data-width=300}  
144 ▾ -----
```

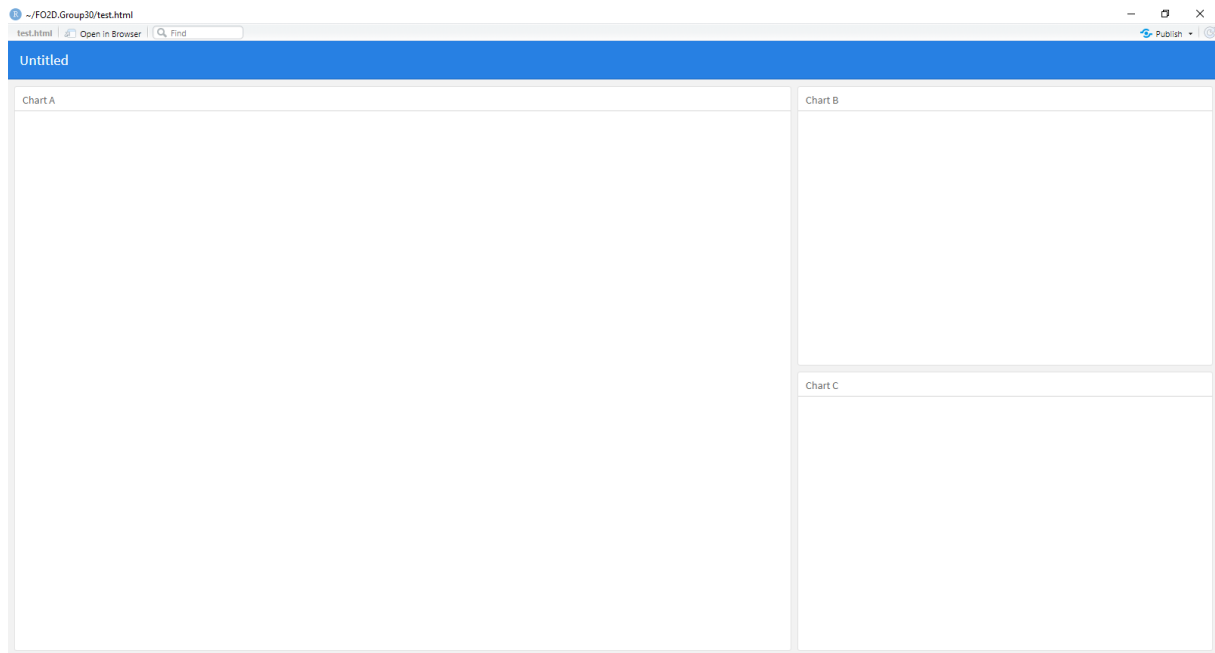
Additionally, one is able to tabs to the columns. This is also established previously by defining the *output* of the dashboard. While *output of scroll* will turn the whole site into a scrollable website, *fill* would make it compact to the column where the data is in. This means that individual columns are scrollable. Since tabs provided a better overview for our project than just a scroll, most of our findings are presented in an interactive manner:

```
column {.tabset .tabset-fade}  
-----
```

{.tabset-fade} is an additional aesthetic argument that makes the transition from one tab to the other more smooth. Tabs are then defined under the Column heading in the form of column names (###).



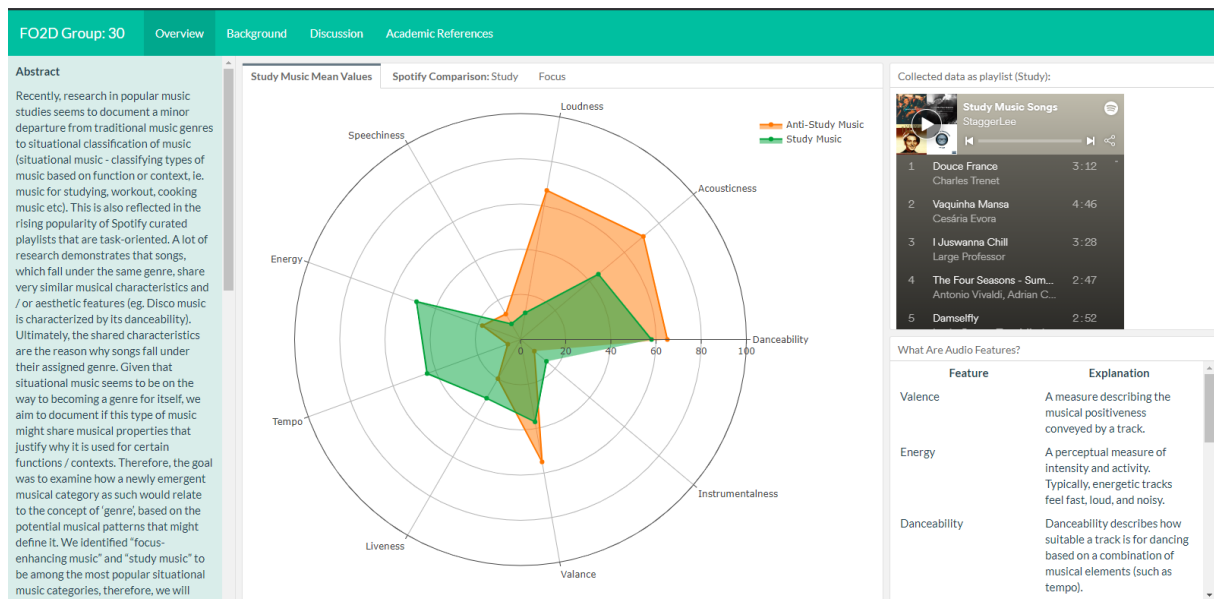
Of course, planning was a good start but is never definite. We were consciously going back and testing different layouts. Each of the name headings would then contain the code for adding chunk of R to each of the tabs respectively.



After trying out numerous versions and modes on how the webpage can look using flexdashboard, we eventually made an outline. The outline was decided based on what information we wanted to present, and more importantly, which information was important to the story we wanted to tell. It took us quite some trials and errors. We were particularly struggling with defining the sized of the column boxes, since sometimes it would not be registered. In addition, the way in which the code that determines the outlines is read was at first trivial, since the output would sometimes result in an unorganized mess; or it would fail to read the data inside.

Dashboard & Visualizations: Code

Overview Page



The purpose of the home page (first page) is to serve as a study abstract, showcasing our main findings and results. It contains a brief introduction and methodology to our study (in the green sidebar) and the overall results. The first graph shows the mean values of the audio features we got from our participants. The two tabs in the same column compare our results to the mean audio features from popular Spotify playlists that are envisioned for the same purpose as the music that we examined. This is so we can provide additional insights to how what that music genre ‘looks like’.

The *Abstract* sidebar was made using a standard markdown heading in flexdashboard:

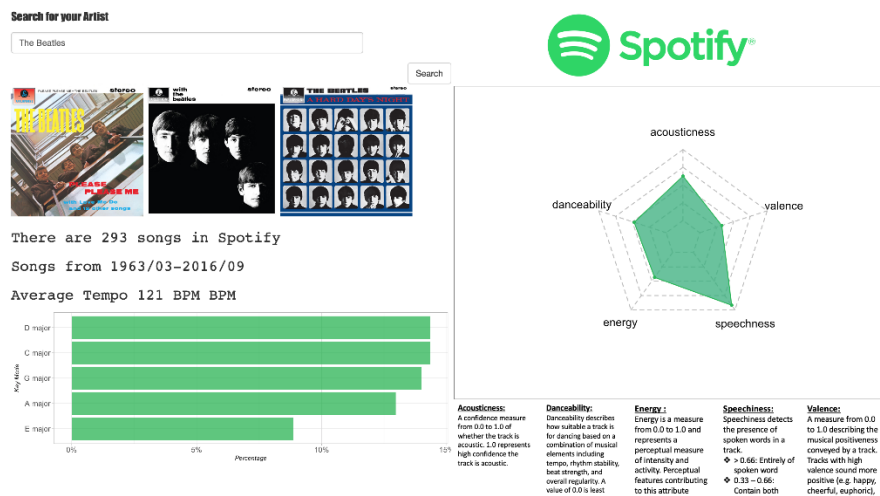
```
13 Overview
14 =====
15
16
17 Sidebar {.sidebar data-width=300}
18 =====
19 **Abstract**
20
21 Recently, research in popular music studies seems to document a minor departure from traditional music genres to situational
22 classification of music (situational music - classifying types of music based on function or context, ie. music for studying,
23 workout, cooking music etc). This is also reflected in the rising popularity of spotify curated playlists that are
24 task-oriented. A lot of research demonstrates that songs, which fall under the same genre, share very similar musical
25 characteristics and / or aesthetic features (eg. Disco music is characterized by its danceability). Ultimately, the shared
26 characteristics are the reason why songs fall under their assigned genre. Given that situational music seems to be on the way to
27 becoming a genre for itself, we aim to document if this type of music might share musical properties that justify why it is used
28 for certain functions / contexts. Therefore, the goal was to examine how a newly emergent musical category as such would relate
29 to the concept of 'genre', based on the potential musical patterns that might define it. We identified "focus-enhancing music"
30 and "study music" to be among the most popular situational music categories, therefore, we will inquire into the characteristics
31 of songs that are used for that function, with the hope of capturing their specific commonalities.
32
33 Through an online survey, we gathered a list of songs that students passively or actively listen to when engaging in
34 study-related activities. In addition, the participants were also asked to supply the name of songs that they would never listen
35 to while studying, or that they believed would obscure their studying process. The musical characteristics of those songs are
36 then tested and visualized through the use of the spotify api.
```

The page is defined by a heading (overview) and separated from the rest using ‘=====’. The other pages in the dashboard are made in the same way. The sidebar comes by default very narrow, therefore we increased its width. Our research text is then written under that heading.

The central graph takes most of the websites front page, since it displays our main results. Determining in what way to visualize it was a dilemma. None of the graphs that we were familiar with were suitable to showcase this number of values in a conceptual form. While radar charts or spider graphs are commonly not preferred, it seemed as it was the best fit for our type of display. In addition,

while researching projects that were conducted using the Spotify API, it became apparent that most users visualize audio features in the same way. Example:

Music Analysis tool with Spotify



Source: <https://towardsdatascience.com/step-by-step-to-visualize-music-genres-with-spotify-api-ce6c273fb827?gi=71fa5a837c6e>

While we had couple of attempts to construct the graphs using different packages, we finally decided to present all our outputs through the *library plotly*. Other common packages, such as: *ggplot* and *ggplot2*; were also tried, but *ggplot2* required more expertise and better dataset management than we had. Whereas *ggplot* would only output very plain graphs that look like they were made in MS Paint. *Plotly* makes use of *Shiny* which is used to build we applications. Therefore, we were able to create graphs that are interactive (i.e. One can hover over them to see certain values and zoom in / out). We did try presenting those finding as bar plots, but they would be more difficult to read and the differences between the values would be less observable.

The “Audio Features Explanation” column is made with the following markdown language from *flexdashboard*:

```
147  
148 ### What Are Audio Features?  
149  
150 Feature | Explanation  
151 ----- | -----  
152 Valence | A measure describing the musical positiveness conveyed by a track.  
153 Energy | A perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.  
154 Danceability | Danceability describes how suitable a track is for dancing based on a combination of musical elements (such as tempo).  
155 Instrumentalness | Predicts whether a track contains no vocals. “ooh” and “aah” sounds are treated as instrumental in this context.  
156 Acoustness | A confidence measure of whether the track is acoustic. 100 represents acoustness, and means there are no electric sounds involved.  
157 Speechiness | Speechiness detects the presence of spoken words in a track.  
158 Liveness | Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.  
159 Loudness | The overall loudness of a track in decibels (dB). values typical range between -60 and 0 db (represented on a scale from 0 - 100)  
160 Tempo | The overall estimated tempo of a track in beats per minute (BPM). Tempo is the speed or pace of a given piece and derives directly from the average beat duration.  
161
```

Again, it is basic markdown language that defines the columns. More importantly, we were met with a frustrating issue here. As explained on Spotifys developer website, the values returned for all the

features are on a scale from 0 to 1. However, Tempo (beats per minute) go up to a few hundreds and the loudness is measured in decibel (from negative 60 to 0). Adding their mean value to the graph with the other variable would result in a non-readable mess (since one of them would take over the whole graph). First, we tried dividing tempo and loudness with a 100 so that they could be on an equal scale. Eventually we realized that this is a way in which one can lie with statistics. While loudness might show its value is 0.24, its actual value is 2400. This was also a problem since the difference between our corpus and anti-corpus would not be representative (i.e. It would look like they are equally loud or equally phased). Finally, a colleague of ours from a more quantitative study explained that we could add 60 to loudness and divide by 60 (e.g. -25 changes to $(-25+60)/60 = 0.583$). This way we change the range between 0 and 1, where 1 is loud, as intuition suggests. Tempo is a bit more complicated. It is measured in beats per minute (BPM), and there is no clear limit to this. However, all of our songs were safely between 60 and 240 BPM, and we decided we would rescale tempo so that 60 becomes 0 and 240 becomes 1. Thus, if a song has 180 BPM, on our scale it would show as 0.67. We are still uncertain if that is the right factorization and if it is adequate to reality. Finally, we decided to bring the whole dataset to a scale from 0 to 100. In turn, this did not change the look of the graph, but it makes the values easier to read.

Referring back to our central visualization, it was made using the following code from Plotly:

```
32 v ### **Study Music Mean Values**
33
34 v {r}
35 library(plotly)
36 library(gridExtra)
37 fig <- plot_ly(
38   type = 'scatterpolar',
39   fill = 'toself',
40   mode = 'lines+markers'
41 )
42 fig <- fig %>%
43   add_trace(
44     r = c(65, 71, 67, 13, 18, 6, 20, 55, 8),
45     theta = c('Danceability', 'Acousticness', 'Loudness', 'Speechiness', 'Energy', 'Tempo', 'Liveness', 'Valance',
46       'Instrumentalness'),
47     name = 'Anti-Study Music'
48   )
49 fig <- fig %>%
50   add_trace(
51     r = c(58, 45, 12, 8, 49, 44, 30, 37, 15),
52     theta = c('Danceability', 'Acousticness', 'Loudness', 'Speechiness', 'Energy', 'Tempo', 'Liveness', 'Valance',
53       'Instrumentalness'),
54     name = 'Study Music'
55   )
56 fig <- fig %>%
57   layout(
58     polar = list(
59       radialaxis = list(
60         visible = T,
61         range = c(0,100)
62       )
63     )
64   )
```

This code is the same for all the radar graphs (that show values for different demographics) on the dashboard. Problems we had with plotly was that it would not let us resize the graphs; add titles (can be added, but look sloppy); and reading our datasets in a data.frame or table format. Therefore, we calculated the mean values for the audio features in a different R script and copy pasted the values in the graph code. The values are added to line 44 (anti-study songs) and line 50 (study songs). Beneath

it, the labels are applied. The position of a numeric value corresponds with the position of the label. Finally, we defined the range of the graph to be from 0 to 100 and added the “*lines+markers*” mode (so that the 2 categories are more distinguishable from each other).

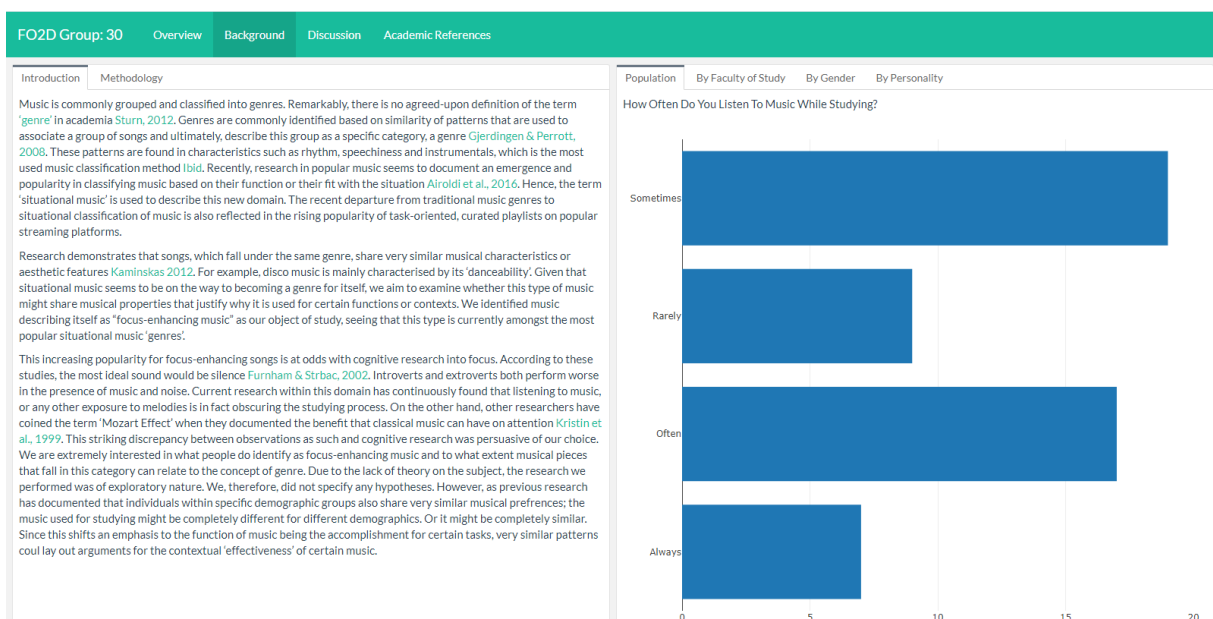
Finally, we used Spotifys Widget tool on its developer page in order to make a “Spotify plat bottom”. This refers to the column in the upper right corner, which shows the songs we collected from our participants as a playable playlist.

```
141
142 column {data-width=300}
143
144 ##### collected data as playlist (study):
145
146 <iframe src="https://open.spotify.com/embed/playlist/5xLS54s1REH7LE2llXT4Id" width="300" height="380" frameborder="0"
147 allowtransparency="true" allow="encrypted-media"></iframe>
```

By having the Spotify IDs and URIs of the songs, a playlist was made using the Spotify Online Console. The URI of the playlist is pasted in the Spotify Widget tool which gave us a 3rd party html to host the playlist. We additionally changed the height and width of it to fit better with the column.

Background Page

This page presents out theoretical framework and methodological approach to our research. The column on the right contains 4 tabs, each of them being a visualization of the demographic groups of the surveyed population.



The first graph is illustrative of how frequently the participants listen to music (passively or actively) while studying (i.e. How often do you listen to music while studying?). While horizontal bar plots are certainly not the best option to present this statistic, we wanted to restrain from using too many of the same graphs (since the goal is to get accustomed to as much as possible). The code for that is presented below.

```

column {.tabset}
-----
### Population

How Often Do You Listen To Music While Studying?

```{r}
library(plotly)

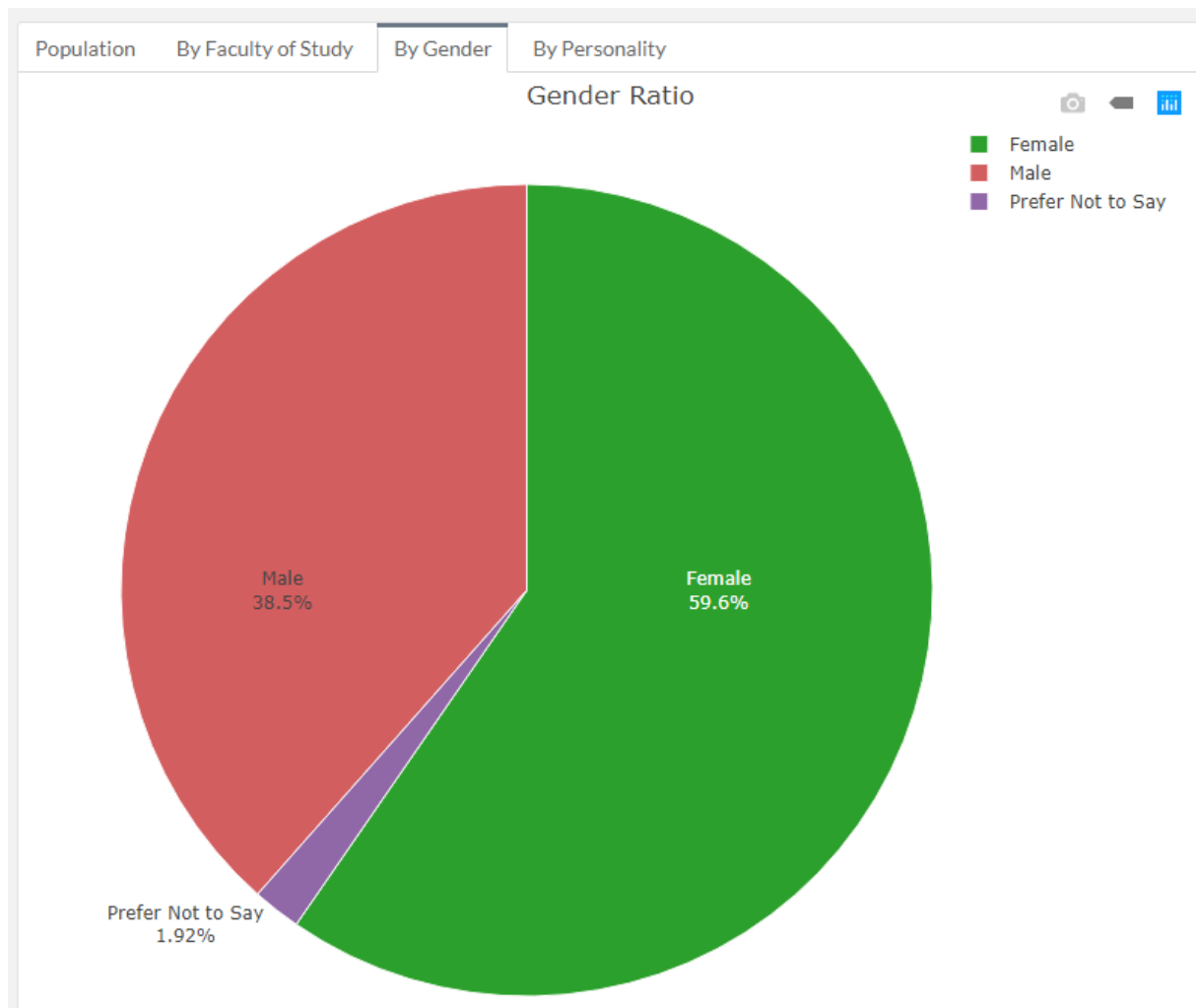
fig <- plot_ly(x = c(3, 9, 19, 17, 7), y = c('Never', 'Rarely', 'Sometimes', 'Often', 'Always'), type = 'bar', orientation =
'h')

fig
```

```

The code is relatively simple. A *plot_ly* call retrieves the functionality from the package, X is used to input the numeric values; and Y to determine the labels. The “*bar*” argument instructs it to present it in a bar plot. Additionally, bar plots are usually vertical, but one can specify orientation to be horizontal. We also tried making every bar into its own color by specifying *color = c(rgb('color pallet identifier'), rgb(...))* but that did not work with a basic bar plot outline. We had other options in mind (also with the use of other packages) but most of them had just too many lines of code. Whilst it is relatively easy to copy-paste the code and change the values / outline / names; there would be relatively little use in that (since we would not understand why and how some of the code works).

Additionally, we grouped our population in male / female and faculty in pie charts. A doughnut chart would be more stylish, but at this stage we were still met with the challenge of accessing .excel and .csv files (this would be resolved, but at the end of our project). We attempted to make a histogram in R, showing how gender is distributed over age, but we did not have the statistical expertise for that. While we did manage to do that with other training datasets, we would not know how to apply it to this case.

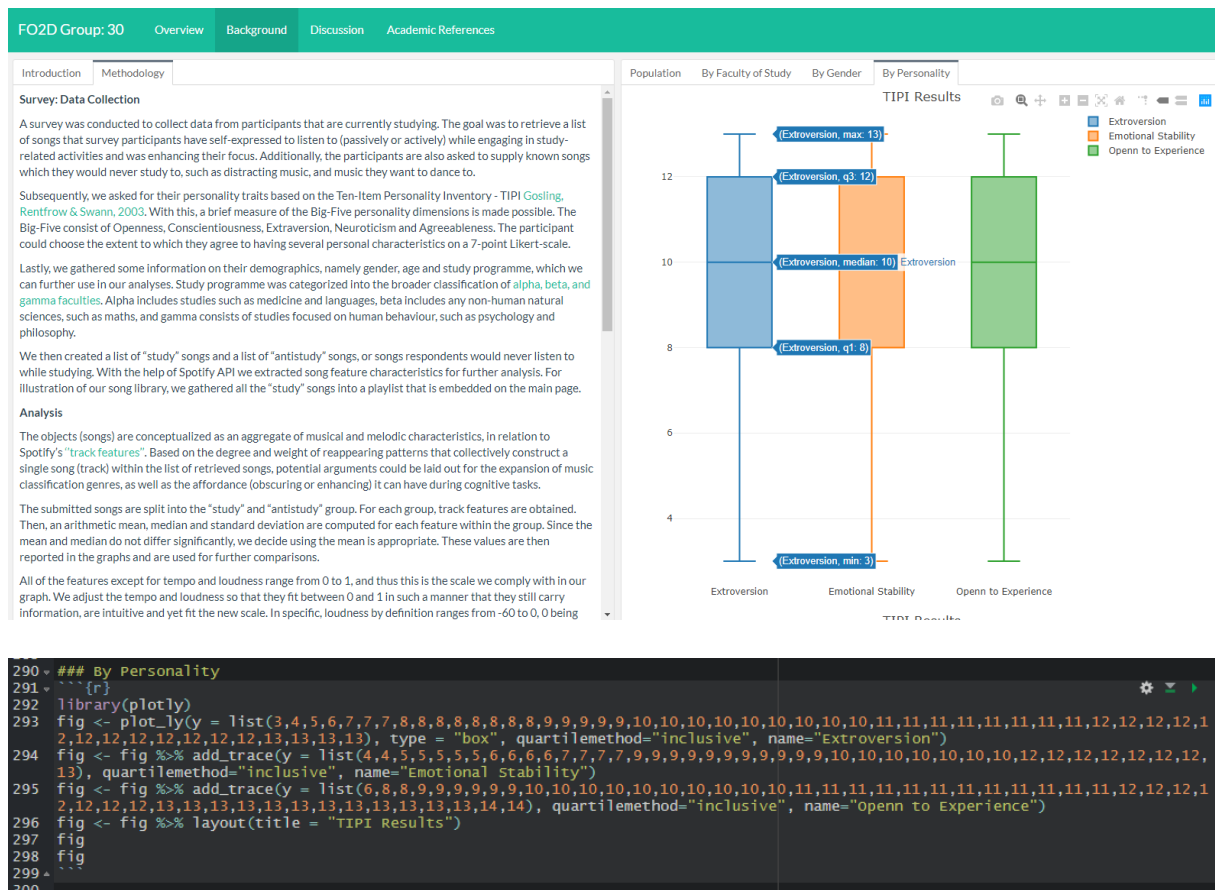


```

258 ## By Gender
259 ```{r}
260 library(plotly)
261 colors <- c('rgb(211,94,96)', 'rgb(44, 160, 44)', 'rgb(144,103,167)', 'rgb(171,104,87)', 'rgb(114,147,203)')
262 labels = c('Male', 'Female', 'Prefer Not to Say')
263 values = c(20, 31, 1)
264 fig <- plot_ly(type='pie', labels=labels, values=values,
265               textinfo='label+percent',
266               insidetextorientation='radial',
267               marker = list(colors = colors,
268                             line = list(color = 'FFFFFF', width = 1)),
269               #The 'pull' attribute can also be used to create space between the sectors
270               showlegend = TRUE)
271 fig <- fig %>% layout(title = 'Gender Ratio',
272                     xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
273                     yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE))
274 fig
275 ```

```

The procedure is vastly similar to the previous one. However, after specifying the values and labels we added `""labels+percent""` call, which turns the values from our data frame into percentages. Next to that, we experimented with the color pallet, but that is about it.



Furthermore, we saw in other research papers that boxplots are commonly used to present the outcomes of a TIPI analysis (personalities). As explained on the beginning of this document, survey participants had to express how they would rank themselves for certain personality traits. The weight of this rank is then factorized, with the final number determining their personal characteristics with regards to extroverted / introverted, emotional stability / instability, openness to new experiences / closeness. Again, since we were still unable to troubleshoot the external-excel-file problem; we imagined what the process of accessing the database would be and eventually entered the values manually after computing them in excel. As we understood it, the function would scan a certain column in the dataset and count and add every numeric value under the heading. The lowest value would be presented on the bottom of the plot, the highest value on top, and the media value in the middle. This makes it straightforward to understand representative descriptive scores of our population. However, we now realize that the graph contains no indicator for the reader to understand where the line is that determines how many people fall under which category (i.e. Extroverted / introverted).

Finally, the "Discussion" page presents our results and related them to the literature introduced in the background. The values for audio features for each demographic are also on that page. The "Academic References" part is another page where we outline literature that was studied and used for

the purpose of this research. Additionally, we were not able to present our raw data in the portfolio since the layout would not read it, for unknown reasons.