Take-Home\_Ex02

Andre Ong Jia Kang

2025-05-23

# Overview

For Take-home Exercise 2, my group members ([Hendra](https://norhendra-isss608.netlify.app/) and [Jin Yao](https://isss608-blessmebellburve.netlify.app/)) and I have decided to do [Mini-Challenge 1](https://vast-challenge.github.io/2025/MC1.html).

Through our discussion (and the help of a wheel randomizer), we split the challenge’s questions amongst ourselves. Hendra will be doing question 2, Jin Yao will be doing question 1 and I will be doing question 3.

## Scenario + Question 3

My task in MC1 is to help Silas, a local journalist to create beautiful and informative visualizations of this data and uncover new and interesting information. Below is the question I am tasked to solve.

**3.Use your visualizations to develop a profile of what it means to be a rising star in the music industry.**

a. *Visualize the careers of three artists. Compare and contrast their rise in popularity and influence.*

b. *Using this characterization, give three predictions of who the next Oceanus Folk stars with be over the next five years.*

## Getting Started

Loading the R packages required.

pacman::p\_load(tidyverse, jsonlite, SmartEDA, tidygraph, ggraph, visNetwork, gridExtra, igraph)

In the code below, ‘fromJSON’ of **jsonlite** package is used to import *MC1\_graph.json* file ino R.

kg <- fromJSON("data/MC1\_graph.json")

### Inspecting sturcture

str(kg, max.level = 1)

List of 5  
 $ directed : logi TRUE  
 $ multigraph: logi TRUE  
 $ graph :List of 2  
 $ nodes :'data.frame': 17412 obs. of 10 variables:  
 $ links :'data.frame': 37857 obs. of 4 variables:

### Extract and Inspect

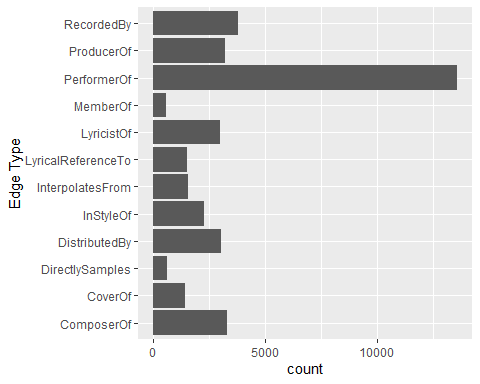
* Data cleaning by splitting the nodes and links.

nodes\_tbl <- as\_tibble(kg$nodes)  
edges\_tb1 <- as\_tibble(kg$links)

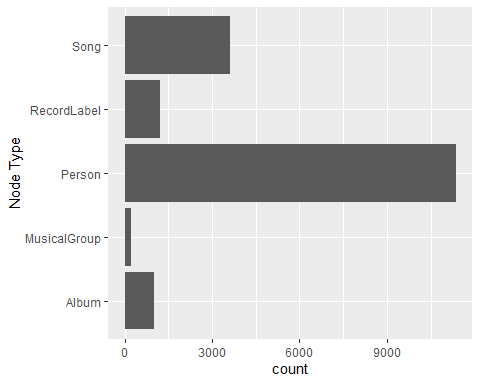
### Initial Data Checks

I will be checking the categories for the Nodes,Edges and their count.

ggplot(data = edges\_tb1,  
 aes(y = `Edge Type`)) + geom\_bar()



ggplot(data = nodes\_tbl,  
 aes(y = `Node Type`)) + geom\_bar()



### visNetwork plot

I will draw up a simple visNetwork to show the linkages of Edge Type == “MemberOf’ to respective Person and MusicalGroup nodes. This is for the initial step for visualization and checking the affliation of each Person to a Group.

# 1. Load & index the graph  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes)  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
id\_map <- nodes\_tbl %>%   
 mutate(idx = row\_number()) %>%   
 select(id, idx)  
  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from), !is.na(to))  
  
# 2. Keep only MemberOf edges and their nodes  
member\_edges <- edges\_idx %>% filter(`Edge Type` == "MemberOf")  
member\_node\_ids <- unique(c(member\_edges$from, member\_edges$to))  
member\_nodes <- nodes\_tbl %>%   
 mutate(idx = row\_number()) %>%   
 filter(idx %in% member\_node\_ids)  
  
# 3. Build the visNetwork nodes df using the original idx as id  
nodes\_df <- member\_nodes %>%  
 transmute(  
 id = idx, # must match edges 'from'/'to'  
 label = name,  
 group = `Node Type`,  
 title = paste0("<b>", name, "</b><br>Type: ", `Node Type`)  
 )  
  
nodes\_df <- nodes\_df %>% arrange(label)  
# 4. Build the edges df using the same idx values  
edges\_df <- member\_edges %>%  
 transmute(  
 from = from, # matches nodes\_df$id  
 to = to, # matches nodes\_df$id  
 title = `Edge Type`  
 )  
  
# 5. Render the network  
visNetwork(nodes\_df, edges\_df, height = "600px", width = "100%") %>%  
 visNodes(shape = "dot", font = list(color = "black")) %>%  
 visEdges(arrows = "to") %>%  
 visOptions(  
 highlightNearest = list(enabled = TRUE, degree = 1),  
 nodesIdSelection = list(enabled = TRUE, useLabels = TRUE)  
 ) %>%  
 visLegend(useGroups = TRUE, width = 0.1, position = "right")



## Data Exploring for Top Artists across all genres

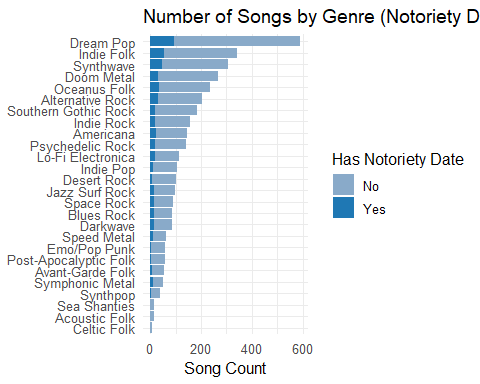
Let’s check for the most popular genres in the music industry. I want to find out which genres have the most released songs and also whether their songs have appeared on a top record chart.

|  |
| --- |
| Nodes to be used |
| • notable (boolean) – whether or not the song has appeared on a top record chart  • notoriety\_date (string) – if provided, the year in which the song first appeared on a top record chart  • genre (string) – the song’s genre |

### Total songs in genres (notoriety\_date)

* Let’s find out the top genres with the most notoriety\_dates.

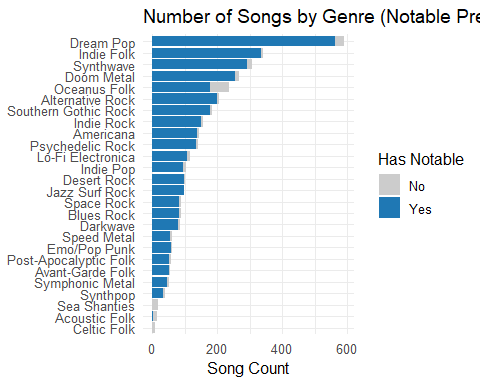
library(gridExtra)  
  
# 1. Load your data  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes)  
  
# 2. Filter to Song nodes and flag notoriety  
songs <- nodes\_tbl %>%   
 filter(`Node Type` == "Song") %>%  
 mutate(  
 has\_notoriety = !is.na(notoriety\_date)  
 )  
  
# 3. Count of songs per genre, broken down by notoriety  
genre\_counts <- songs %>%  
 count(genre, has\_notoriety, name = "Count") %>%  
 arrange(genre)  
  
p1 <- ggplot(genre\_counts, aes(x = fct\_reorder(genre, Count, .fun = sum),   
 y = Count,   
 fill = has\_notoriety)) +  
 geom\_col() +  
 coord\_flip() +  
 scale\_fill\_manual(  
 name = "Has Notoriety Date",  
 values = c(`TRUE` = "#1f78b4", `FALSE` = "#89aac9"),  
 labels = c("No", "Yes")  
 ) +  
 labs(  
 title = "Number of Songs by Genre (Notoriety Date Present)",  
 x = NULL,  
 y = "Song Count"  
 ) +  
 theme\_minimal(base\_size = 12)  
  
# 4. Optional: Average popularity per genre (if available)  
has\_pop <- "popularity" %in% colnames(songs)  
if (has\_pop) {  
 genre\_pop <- songs %>%  
 group\_by(genre) %>%  
 summarise(AvgPopularity = mean(popularity, na.rm = TRUE),  
 .groups = "drop") %>%  
 arrange(desc(AvgPopularity))  
   
 p2 <- ggplot(genre\_pop, aes(x = fct\_reorder(genre, AvgPopularity), y = AvgPopularity)) +  
 geom\_col(fill = "tomato") +  
 coord\_flip() +  
 labs(  
 title = "Average Song Popularity by Genre",  
 x = NULL,  
 y = "Avg. Popularity"  
 ) +  
 theme\_minimal(base\_size = 12)  
   
 # 5. Display side-by-side  
 grid.arrange(p1, p2, ncol = 2)  
} else {  
 # 5. If no popularity field, just show the stacked count plot  
 print(p1)  
}



### Total songs in genres (notoble)

* We get a much better representation of the total songs an whether it appeared on the top charts below.
* Dream Pop has the most songs while Celtic Folk has the least amount of songs in the dataset throughout the years.

library(gridExtra)  
  
# 1. Load your data  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes)  
  
# 2. Filter to Song nodes and flag notable  
songs <- nodes\_tbl %>%   
 filter(`Node Type` == "Song") %>%  
 mutate(  
 has\_notable = (notable == TRUE)  
 )  
  
# 3. Count of songs per genre, broken down by notable  
genre\_counts <- songs %>%  
 count(genre, has\_notable, name = "Count") %>%  
 arrange(genre)  
  
p1 <- ggplot(genre\_counts, aes(  
 x = fct\_reorder(genre, Count, .fun = sum),   
 y = Count,   
 fill = has\_notable  
 )) +  
 geom\_col() +  
 coord\_flip() +  
 scale\_fill\_manual(  
 name = "Has Notable",  
 values = c(`TRUE` = "#1f78b4",   
 `FALSE` = "grey80"),  
 labels = c("No", "Yes")  
 ) +  
 labs(  
 title = "Number of Songs by Genre (Notable Present)",  
 x = NULL,  
 y = "Song Count"  
 ) +  
 theme\_minimal(base\_size = 12)  
  
# 4. Optional: Average popularity per genre (if available)  
has\_pop <- "popularity" %in% colnames(songs)  
if (has\_pop) {  
 genre\_pop <- songs %>%  
 group\_by(genre) %>%  
 summarise(AvgPopularity = mean(popularity, na.rm = TRUE),  
 .groups = "drop") %>%  
 arrange(desc(AvgPopularity))  
   
 p2 <- ggplot(genre\_pop, aes(  
 x = fct\_reorder(genre, AvgPopularity),   
 y = AvgPopularity  
 )) +  
 geom\_col(fill = "tomato") +  
 coord\_flip() +  
 labs(  
 title = "Average Song Popularity by Genre",  
 x = NULL,  
 y = "Avg. Popularity"  
 ) +  
 theme\_minimal(base\_size = 12)  
   
 # 5. Display side-by-side  
 grid.arrange(p1, p2, ncol = 2)  
} else {  
 # 5. If no popularity field, just show the stacked count plot  
 print(p1)  
}



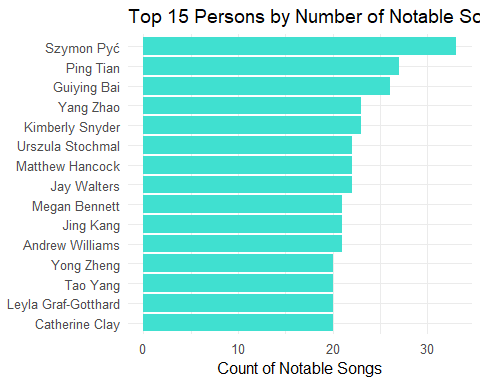
|  |
| --- |
| Note |
| After analysis, there are songs that never get a notoriety\_date chart but are still marked as notable == True. Thus, it would be better to count the notable songs instead of notoriety\_date. |

## Top 15 artists with the most notable songs and if it links to the most popular genres.

* Let’s find out the top 15 artists with the most notable songs in this dataset.

### Top 15 Person by Total of Notable Songs

# 1. Load & index the graph  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>%   
 mutate(idx = row\_number()) # add an index for joining  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
id\_map <- nodes\_tbl %>% select(id, idx)  
  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from) & !is.na(to))  
  
# 2. Identify all Person nodes  
person\_idxs <- which(nodes\_tbl$`Node Type` == "Person")  
  
# 3. Count songs marked 'notable' per Person  
notable\_tbl <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf", "PerformerOf"),  
 from %in% person\_idxs) %>%  
 # join to get each target song's 'notable' flag  
 left\_join(  
 nodes\_tbl %>% select(idx, notable),  
 by = c("to" = "idx")  
 ) %>%  
 # keep only those with notable == TRUE  
 filter(notable == TRUE) %>%  
 mutate(Person = nodes\_tbl$name[from]) %>%  
 count(Person, name = "NotableSongs") %>%  
 arrange(desc(NotableSongs))  
  
# 4. Plot the top 5 Persons by number of notable songs  
top\_n <- 15  
notable\_tbl %>%  
 slice\_head(n = top\_n) %>%  
 ggplot(aes(x = fct\_reorder(Person, NotableSongs), y = NotableSongs)) +  
 geom\_col(fill = "turquoise") +  
 coord\_flip() +  
 labs(  
 title = "Top 15 Persons by Number of Notable Songs",  
 x = NULL,  
 y = "Count of Notable Songs"  
 ) +  
 theme\_minimal(base\_size = 12)

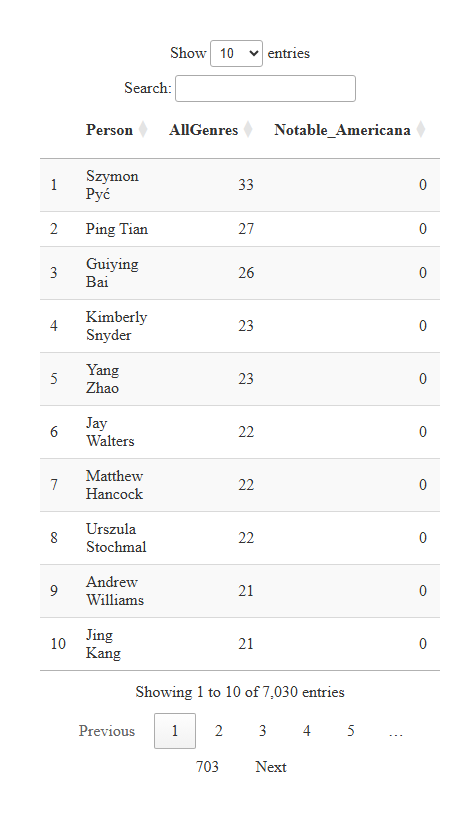


|  |
| --- |
| Plot 6 Note |
| With this, I will like to visualize these 15 top performers and their affiliation to each genres just as a preview. |

### Table of Artist and Genres appeared

* The table below shows the notable songs of the artists into the selected top 10 genres.
* **Top 10 Genres based on total number of songs:**
* “Dream Pop”, “Indie Folk”, “Synthwave”, “Doom Metal”, “Oceanus Folk”, “Alternative Rock”, “Southern Gothic Rock”, “Indie Rock”, “Americana”, “Psychedelic Rock”

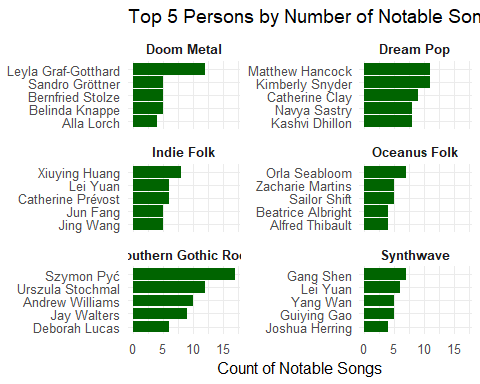
# 1. Compute total notable songs per Person  
notable\_tbl\_edit <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"),  
 from %in% person\_idxs) %>%  
 left\_join(nodes\_tbl %>% select(idx, notable), by = c("to" = "idx")) %>%  
 filter(notable == TRUE) %>%  
 mutate(Person = nodes\_tbl$name[from]) %>%  
 count(Person, name = "NotableSongs") %>%  
 arrange(desc(NotableSongs))  
  
# 2. Define the 10 genres of interest  
genres <- c(  
 "Dream Pop", "Indie Folk", "Synthwave", "Doom Metal", "Oceanus Folk",  
 "Alternative Rock", "Southern Gothic Rock", "Indie Rock",  
 "Americana", "Psychedelic Rock"  
)  
  
# 3. Compute notable‐song counts by Person × Genre  
genre\_counts <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"),  
 from %in% person\_idxs) %>%  
 left\_join(  
 nodes\_tbl %>%   
 filter(`Node Type` == "Song", genre %in% genres, notable == TRUE) %>%  
 select(idx, genre),  
 by = c("to" = "idx")  
 ) %>%  
 filter(!is.na(genre)) %>%  
 mutate(Person = nodes\_tbl$name[from]) %>%  
 count(Person, genre, name = "Count")  
  
# 4. Pivot those genres into columns for all persons  
comparison <- notable\_tbl\_edit %>%  
 rename(AllGenres = NotableSongs) %>%  
 left\_join(  
 genre\_counts %>%  
 pivot\_wider(  
 id\_cols = Person,  
 names\_from = genre,  
 values\_from = Count,  
 values\_fill = 0,  
 names\_prefix = "Notable\_"  
 ),  
 by = "Person"  
 ) %>%  
 # fill any missing genre columns with zeros  
 mutate(across(where(is.numeric), ~replace\_na(.x, 0))) %>%  
 # reorder: Person, AllGenres, then the Notable\_\* genres  
 select(Person, AllGenres, starts\_with("Notable\_"))  
  
# 5. Display as a horizontally scrollable DataTable  
library(DT)  
DT::datatable(  
 comparison,  
 options = list(  
 scrollX = TRUE,  
 pageLength = 10  
 )  
)



### Top persons in the top 6 genres based on notarable songs.

I will select the top 6 genres based on the previous plot and filter to the top 5 artists with the most notable songs.

# 1. Load & index the graph  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>%   
 mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from) & !is.na(to))  
  
# 2. Identify all Person nodes  
person\_idxs <- which(nodes\_tbl$`Node Type` == "Person")  
  
# 3. Define your genres of interest  
genres <- c("Dream Pop", "Indie Folk", "Synthwave", "Doom Metal", "Oceanus Folk", "Southern Gothic Rock")  
  
# 4. Build a lookup of song‐idx → genre + notable flag, for these genres  
genre\_songs <- nodes\_tbl %>%  
 filter(`Node Type` == "Song", genre %in% genres) %>%  
 select(idx, genre, notable)  
  
# 5. Count notable songs per Person per genre  
notable\_genre\_tbl <- edges\_idx %>%  
 filter(  
 `Edge Type` %in% c("ComposerOf", "PerformerOf"),  
 from %in% person\_idxs  
 ) %>%  
 inner\_join(genre\_songs, by = c("to" = "idx")) %>%  
 filter(notable == TRUE) %>%  
 mutate(Person = nodes\_tbl$name[from]) %>%  
 count(genre, Person, name = "NotableSongs") %>%  
 arrange(genre, desc(NotableSongs))  
  
# 6. Take top 5 Persons in each genre  
top5\_by\_genre <- notable\_genre\_tbl %>%  
 group\_by(genre) %>%  
 slice\_max(NotableSongs, n = 5, with\_ties = FALSE) %>%  
 ungroup()  
  
# 7. Plot small multiples  
ggplot(top5\_by\_genre,   
 aes(x = fct\_reorder(Person, NotableSongs), y = NotableSongs)) +  
 geom\_col(fill = "darkgreen") +  
 coord\_flip() +  
 facet\_wrap(~ genre, scales = "free\_y", ncol = 2) +  
 labs(  
 title = "Top 5 Persons by Number of Notable Songs in select 6 most popular Genres",  
 x = NULL,  
 y = "Count of Notable Songs"  
 ) +  
 theme\_minimal(base\_size = 12) +  
 theme(strip.text = element\_text(face = "bold"))



|  |
| --- |
| Selection |
| Based on the charts shown, I will choose the following artist based on high number of notable songs and popular genre. I will also explore and derive some findings of the 3 artists and any characteristics on how **popular** & **influential** they are as top artist in their field.  **1. Kimberly Snyder**  **2. Leyla Graf-Gotthard**  **3. Szymon Pyć** |

# Question 3 Part (a)

Part (a): Visualize the careers of three artists. Compare and contrast their rise in popularity and influence

I have picked the following three artists and genres based on the data exploring conducted:

*1. Kimberly Snyder - Dream Pop*

*2. Leyla Graf-Gotthard - Doom Metal*

*3. Szymon Pyć - Southern Gothic Rock*

I will build a tiny visNetwork graph showing their releases and collaborations over time to check on their popularity and influence.

## Popularity: Comparison Overall vs in Genre by Notable Songs

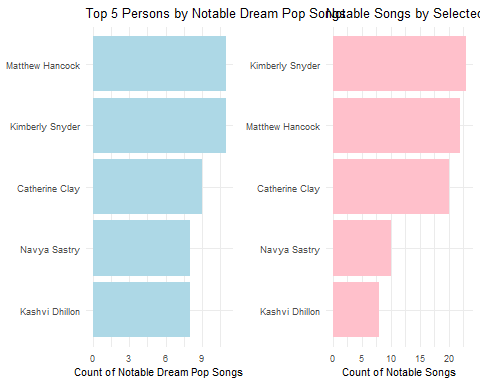
Using notable songs as a proxy for popularity, let’s compare the total notable songs of the selected 3 artists in their respective genre vs overall.

This will show how many notable songs the artist produced were actually their main genre.

### Kimberly Snyder

* Based on the bar graphs, Kimberly Snyder has the most notable songs released but ties with Matthew Hancock in the Dream Pop genre. However, this also shows that Kimberly has influence on other genres.

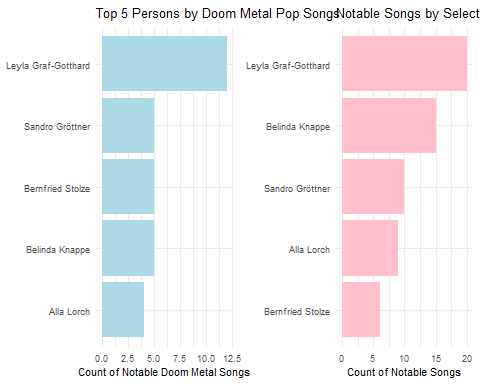
library(gridExtra)  
  
# 1. Load & index the graph  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>% mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from) & !is.na(to))  
  
person\_idxs <- which(nodes\_tbl$`Node Type` == "Person")  
  
# 2. Dream Pop: top 5 Persons by notable Dream Pop songs  
dp\_song\_idxs <- nodes\_tbl %>%  
 filter(`Node Type` == "Song", genre == "Dream Pop") %>%  
 pull(idx)  
  
notable\_dp\_tbl <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"),  
 from %in% person\_idxs, to %in% dp\_song\_idxs) %>%  
 left\_join(nodes\_tbl %>% select(idx, notable), by = c("to" = "idx")) %>%  
 filter(notable == TRUE) %>%  
 mutate(Person = nodes\_tbl$name[from]) %>%  
 count(Person, name = "NotableDreamPopSongs") %>%  
 arrange(desc(NotableDreamPopSongs))  
  
p1 <- notable\_dp\_tbl %>%  
 slice\_head(n = 5) %>%  
 ggplot(aes(x = fct\_reorder(Person, NotableDreamPopSongs),   
 y = NotableDreamPopSongs)) +  
 geom\_col(fill = "lightblue") +  
 coord\_flip() +  
 labs(  
 title = "Top 5 Persons by Notable Dream Pop Songs",  
 x = NULL,  
 y = "Count of Notable Dream Pop Songs"  
 ) +  
 theme\_minimal(base\_size = 10)+  
 theme(  
 plot.title = element\_text(size = 10),  
 plot.subtitle= element\_text(size = 8),  
 axis.title.x = element\_text(size = 8),  
 axis.text = element\_text(size = 7)  
 )  
  
# 3. Selected Persons: overall notable songs  
focus\_people <- c("Kimberly Snyder", "Matthew Hancock",   
 "Catherine Clay", "Navya Sastry", "Kashvi Dhillon")  
  
notable\_tbl <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"),  
 from %in% person\_idxs) %>%  
 left\_join(nodes\_tbl %>% select(idx, notable), by = c("to" = "idx")) %>%  
 filter(notable == TRUE) %>%  
 mutate(Person = nodes\_tbl$name[from]) %>%  
 count(Person, name = "NotableSongs") %>%  
 arrange(desc(NotableSongs))  
  
plot\_tbl <- tibble(Person = focus\_people) %>%  
 left\_join(notable\_tbl, by = "Person") %>%  
 replace\_na(list(NotableSongs = 0))  
  
p2 <- plot\_tbl %>%  
 ggplot(aes(x = fct\_reorder(Person, NotableSongs), y = NotableSongs)) +  
 geom\_col(fill = "pink") +  
 coord\_flip() +  
 labs(  
 title = "Notable Songs by Selected Persons",  
 x = NULL,  
 y = "Count of Notable Songs"  
 ) +  
 theme\_minimal(base\_size = 10)+  
 theme(  
 plot.title = element\_text(size = 10),  
 plot.subtitle= element\_text(size = 8),  
 axis.title.x = element\_text(size = 8),  
 axis.text = element\_text(size = 7)  
 )  
  
# 4. Arrange side‐by‐side  
grid.arrange(p1, p2, ncol = 2)



### Leyla Graf-Gotthard

* Leyla Graf-Gotthard is the top in her genre with some influence in other genres too based on overall.

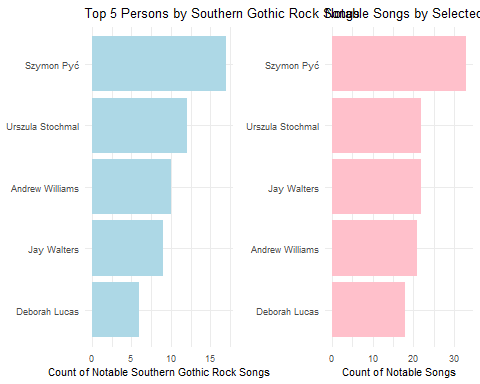
# 2. Dream Pop: top 5 Persons by notable Doom Metal songs  
dp\_song\_idxs2 <- nodes\_tbl %>%  
 filter(`Node Type` == "Song", genre == "Doom Metal") %>%  
 pull(idx)  
  
notable\_dp\_tbl2 <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"),  
 from %in% person\_idxs, to %in% dp\_song\_idxs2) %>%  
 left\_join(nodes\_tbl %>% select(idx, notable), by = c("to" = "idx")) %>%  
 filter(notable == TRUE) %>%  
 mutate(Person = nodes\_tbl$name[from]) %>%  
 count(Person, name = "NotableDoomMetalSongs") %>%  
 arrange(desc(NotableDoomMetalSongs))  
  
p3 <- notable\_dp\_tbl2 %>%  
 slice\_head(n = 5) %>%  
 ggplot(aes(x = fct\_reorder(Person, NotableDoomMetalSongs),   
 y = NotableDoomMetalSongs)) +  
 geom\_col(fill = "lightblue") +  
 coord\_flip() +  
 labs(  
 title = "Top 5 Persons by Doom Metal Pop Songs",  
 x = NULL,  
 y = "Count of Notable Doom Metal Songs"  
 ) +  
 theme\_minimal(base\_size = 10)+  
 theme(  
 plot.title = element\_text(size = 10),  
 plot.subtitle= element\_text(size = 8),  
 axis.title.x = element\_text(size = 8),  
 axis.text = element\_text(size = 7)  
 )  
  
# 3. Selected Persons: overall notable songs  
focus\_people2 <- c("Leyla Graf-Gotthard", "Sandro Gröttner",   
 "Bernfried Stolze", "Belinda Knappe", "Alla Lorch")  
  
notable\_tbl2 <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"),  
 from %in% person\_idxs) %>%  
 left\_join(nodes\_tbl %>% select(idx, notable), by = c("to" = "idx")) %>%  
 filter(notable == TRUE) %>%  
 mutate(Person = nodes\_tbl$name[from]) %>%  
 count(Person, name = "NotableSongs") %>%  
 arrange(desc(NotableSongs))  
  
plot\_tbl2 <- tibble(Person = focus\_people2) %>%  
 left\_join(notable\_tbl2, by = "Person") %>%  
 replace\_na(list(NotableSongs = 0))  
  
p4 <- plot\_tbl2 %>%  
 ggplot(aes(x = fct\_reorder(Person, NotableSongs), y = NotableSongs)) +  
 geom\_col(fill = "pink") +  
 coord\_flip() +  
 labs(  
 title = "Notable Songs by Selected Persons",  
 x = NULL,  
 y = "Count of Notable Songs"  
 ) +  
 theme\_minimal(base\_size = 10)+  
 theme(  
 plot.title = element\_text(size = 10),  
 plot.subtitle= element\_text(size = 8),  
 axis.title.x = element\_text(size = 8),  
 axis.text = element\_text(size = 7)  
 )  
  
# 4. Arrange side‐by‐side  
grid.arrange(p3, p4, ncol = 2)



### Szymon Pyć

* Similarly, Szymon is the top in his genre with some influence in other genres too based on overall. He has the most number of notable songs overall.

# 2. Dream Pop: top 5 Persons by notable Dream Pop songs  
dp\_song\_idxs3 <- nodes\_tbl %>%  
 filter(`Node Type` == "Song", genre == "Southern Gothic Rock") %>%  
 pull(idx)  
  
notable\_dp\_tbl3 <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"),  
 from %in% person\_idxs, to %in% dp\_song\_idxs3) %>%  
 left\_join(nodes\_tbl %>% select(idx, notable), by = c("to" = "idx")) %>%  
 filter(notable == TRUE) %>%  
 mutate(Person = nodes\_tbl$name[from]) %>%  
 count(Person, name = "NotableSouthernGothicRockSongs") %>%  
 arrange(desc(NotableSouthernGothicRockSongs))  
  
p5 <- notable\_dp\_tbl3 %>%  
 slice\_head(n = 5) %>%  
 ggplot(aes(x = fct\_reorder(Person, NotableSouthernGothicRockSongs),   
 y = NotableSouthernGothicRockSongs)) +  
 geom\_col(fill = "lightblue") +  
 coord\_flip() +  
 labs(  
 title = "Top 5 Persons by Southern Gothic Rock Songs",  
 x = NULL,  
 y = "Count of Notable Southern Gothic Rock Songs"  
 ) +  
 theme\_minimal(base\_size = 10)+  
 theme(  
 plot.title = element\_text(size = 10),  
 plot.subtitle= element\_text(size = 8),  
 axis.title.x = element\_text(size = 8),  
 axis.text = element\_text(size = 7)  
 )  
  
# 3. Selected Persons: overall notable songs  
focus\_people3 <- c("Szymon Pyć", "Urszula Stochmal","Andrew Williams","Jay Walters","Deborah Lucas")  
  
notable\_tbl3 <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"),  
 from %in% person\_idxs) %>%  
 left\_join(nodes\_tbl %>% select(idx, notable), by = c("to" = "idx")) %>%  
 filter(notable == TRUE) %>%  
 mutate(Person = nodes\_tbl$name[from]) %>%  
 count(Person, name = "NotableSongs") %>%  
 arrange(desc(NotableSongs))  
  
plot\_tbl3 <- tibble(Person = focus\_people3) %>%  
 left\_join(notable\_tbl2, by = "Person") %>%  
 replace\_na(list(NotableSongs = 0))  
  
p6 <- plot\_tbl3 %>%  
 ggplot(aes(x = fct\_reorder(Person, NotableSongs), y = NotableSongs)) +  
 geom\_col(fill = "pink") +  
 coord\_flip() +  
 labs(  
 title = "Notable Songs by Selected Persons",  
 x = NULL,  
 y = "Count of Notable Songs"  
 ) +  
 theme\_minimal(base\_size = 10)+  
 theme(  
 plot.title = element\_text(size = 10),  
 plot.subtitle= element\_text(size = 8),  
 axis.title.x = element\_text(size = 8),  
 axis.text = element\_text(size = 7)  
 )  
  
# 4. Arrange side‐by‐side  
grid.arrange(p5, p6, ncol = 2)



|  |
| --- |
| Notable Songs by Top 5 persons in Genre vs Overall |
| * Based on the notable songs, the selected 3 artists are the top for their genre with influences in other genres as well. This shows the popularity based on using “Notable Songs” as a proxy. |

## Popularity: Notable and Notoriety\_Date plotted against release\_date

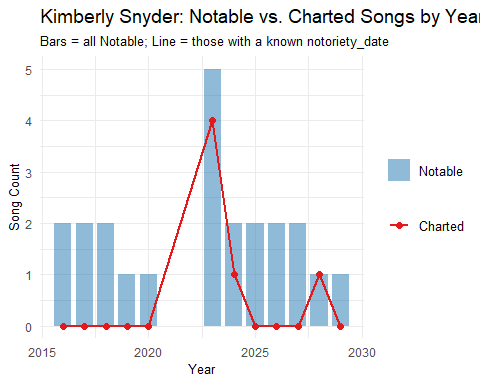
Let’s take a look at the timeline of notable songs for the 3 artists. I will be using notable, release\_date and notoriety\_date for the respective plots.

This will give a broad “popularity over time” view of an artist’s impact.

|  |
| --- |
| Note on Nodes used |
| Below is some explanation on the nodes used for the plots.   * Notable: “this song was popular/critically acclaimed,” regardless of whether we know its exact chart date. * Notoriety\_date: only exists for the subset of notable tracks that actually landed on the charts. It will just be a marker for precised chart‐entry timing information (not the whole thing) as it will always under-count the popularity signal. |

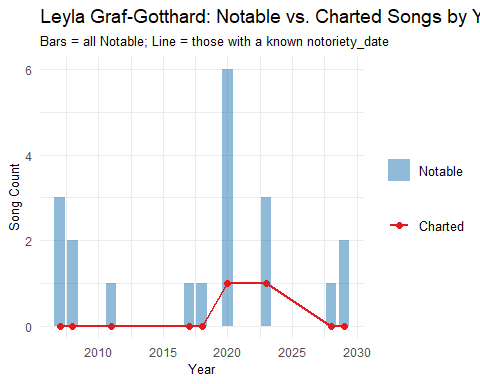
### Kimberly Snyder

# 1. Load & index the graph  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>% mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
# 2. Build edge index  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from), !is.na(to))  
  
# 3. Pull Kimberly Snyder’s song edges  
kim\_idx <- which(nodes\_tbl$name == "Kimberly Snyder")  
kim\_songs <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"), from == kim\_idx) %>%  
 left\_join(  
 nodes\_tbl %>% select(idx, notable, release\_date, notoriety\_date),  
 by = c("to" = "idx")  
 )  
  
# 4. Extract years  
kim\_songs <- kim\_songs %>%  
 mutate(  
 release\_year = str\_extract(release\_date, "\\d{4}") %>% as.integer(),  
 notoriety\_year = str\_extract(notoriety\_date, "\\d{4}") %>% as.integer()  
 )  
  
# 5. Aggregate counts  
notable\_yearly <- kim\_songs %>%  
 filter(notable == TRUE, !is.na(release\_year)) %>%  
 count(Year = release\_year, name = "Notable")   
  
charted\_yearly <- kim\_songs %>%  
 filter(!is.na(notoriety\_year)) %>%  
 count(Year = notoriety\_year, name = "Charted")  
  
combined <- full\_join(notable\_yearly, charted\_yearly, by = "Year") %>%  
 replace\_na(list(Notable = 0, Charted = 0)) %>%  
 arrange(Year)  
  
# 6. Plot both metrics  
ggplot(combined, aes(x = Year)) +  
 geom\_col(aes(y = Notable, fill = "Notable"), alpha = 0.5, width = 0.8) +  
 geom\_line(aes(y = Charted, color = "Charted"), size = 1) +  
 geom\_point(aes(y = Charted, color = "Charted"), size = 2) +  
 scale\_fill\_manual(  
 name = NULL,  
 values = c(Notable = "#1f78b4")  
 ) +  
 scale\_color\_manual(  
 name = NULL,  
 values = c(Charted = "#e31a1c")  
 ) +  
 labs(  
 title = "Kimberly Snyder: Notable vs. Charted Songs by Year",  
 subtitle = "Bars = all Notable; Line = those with a known notoriety\_date",  
 x = "Year",  
 y = "Song Count"  
 ) +  
 theme\_minimal(base\_size = 12) +  
 theme(  
 plot.title = element\_text(size = 14),  
 plot.subtitle = element\_text(size = 10),  
 axis.text = element\_text(size = 9),  
 axis.title = element\_text(size = 10)  
 )



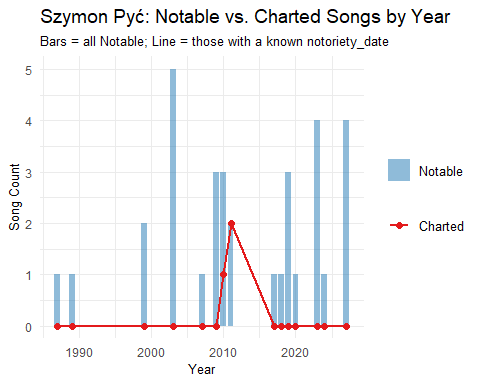
### Leyla Graf-Gotthard

# 3. Pull Kimberly Snyder’s song edges  
leyla\_idx <- which(nodes\_tbl$name == "Leyla Graf-Gotthard")  
leyla\_songs <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"), from == leyla\_idx) %>%  
 left\_join(  
 nodes\_tbl %>% select(idx, notable, release\_date, notoriety\_date),  
 by = c("to" = "idx")  
 )  
  
# 4. Extract years  
leyla\_songs <- leyla\_songs %>%  
 mutate(  
 release\_year = str\_extract(release\_date, "\\d{4}") %>% as.integer(),  
 notoriety\_year = str\_extract(notoriety\_date, "\\d{4}") %>% as.integer()  
 )  
  
# 5. Aggregate counts  
notable\_yearly2 <- leyla\_songs %>%  
 filter(notable == TRUE, !is.na(release\_year)) %>%  
 count(Year = release\_year, name = "Notable")   
  
charted\_yearly2 <- leyla\_songs %>%  
 filter(!is.na(notoriety\_year)) %>%  
 count(Year = notoriety\_year, name = "Charted")  
  
combined2 <- full\_join(notable\_yearly2, charted\_yearly2, by = "Year") %>%  
 replace\_na(list(Notable = 0, Charted = 0)) %>%  
 arrange(Year)  
  
# 6. Plot both metrics  
ggplot(combined2, aes(x = Year)) +  
 geom\_col(aes(y = Notable, fill = "Notable"), alpha = 0.5, width = 0.8) +  
 geom\_line(aes(y = Charted, color = "Charted"), size = 1) +  
 geom\_point(aes(y = Charted, color = "Charted"), size = 2) +  
 scale\_fill\_manual(  
 name = NULL,  
 values = c(Notable = "#1f78b4")  
 ) +  
 scale\_color\_manual(  
 name = NULL,  
 values = c(Charted = "#e31a1c")  
 ) +  
 labs(  
 title = "Leyla Graf-Gotthard: Notable vs. Charted Songs by Year",  
 subtitle = "Bars = all Notable; Line = those with a known notoriety\_date",  
 x = "Year",  
 y = "Song Count"  
 ) +  
 theme\_minimal(base\_size = 12) +  
 theme(  
 plot.title = element\_text(size = 14),  
 plot.subtitle = element\_text(size = 10),  
 axis.text = element\_text(size = 9),  
 axis.title = element\_text(size = 10)  
 )



### Szymon Pyć

# 3. Pull Kimberly Snyder’s song edges  
szymon\_idx <- which(nodes\_tbl$name == "Szymon Pyć")  
szymon\_songs <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"), from == szymon\_idx) %>%  
 left\_join(  
 nodes\_tbl %>% select(idx, notable, release\_date, notoriety\_date),  
 by = c("to" = "idx")  
 )  
  
# 4. Extract years  
szymon\_songs <- szymon\_songs %>%  
 mutate(  
 release\_year = str\_extract(release\_date, "\\d{4}") %>% as.integer(),  
 notoriety\_year = str\_extract(notoriety\_date, "\\d{4}") %>% as.integer()  
 )  
  
# 5. Aggregate counts  
notable\_yearly3 <- szymon\_songs %>%  
 filter(notable == TRUE, !is.na(release\_year)) %>%  
 count(Year = release\_year, name = "Notable")   
  
charted\_yearly3 <- szymon\_songs %>%  
 filter(!is.na(notoriety\_year)) %>%  
 count(Year = notoriety\_year, name = "Charted")  
  
combined3 <- full\_join(notable\_yearly3, charted\_yearly3, by = "Year") %>%  
 replace\_na(list(Notable = 0, Charted = 0)) %>%  
 arrange(Year)  
  
# 6. Plot both metrics  
ggplot(combined3, aes(x = Year)) +  
 geom\_col(aes(y = Notable, fill = "Notable"), alpha = 0.5, width = 0.8) +  
 geom\_line(aes(y = Charted, color = "Charted"), size = 1) +  
 geom\_point(aes(y = Charted, color = "Charted"), size = 2) +  
 scale\_fill\_manual(  
 name = NULL,  
 values = c(Notable = "#1f78b4")  
 ) +  
 scale\_color\_manual(  
 name = NULL,  
 values = c(Charted = "#e31a1c")  
 ) +  
 labs(  
 title = "Szymon Pyć: Notable vs. Charted Songs by Year",  
 subtitle = "Bars = all Notable; Line = those with a known notoriety\_date",  
 x = "Year",  
 y = "Song Count"  
 ) +  
 theme\_minimal(base\_size = 12) +  
 theme(  
 plot.title = element\_text(size = 14),  
 plot.subtitle = element\_text(size = 10),  
 axis.text = element\_text(size = 9),  
 axis.title = element\_text(size = 10)  
 )



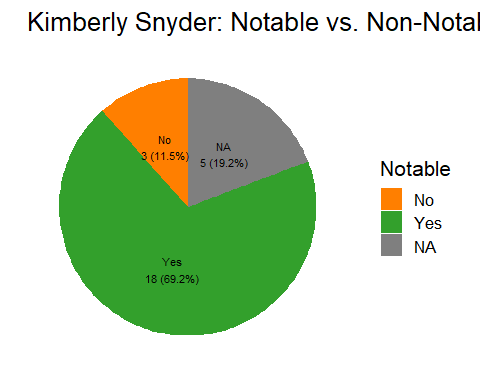
|  |
| --- |
| Notable Songs and Charted Songs across the Years |
| * We can see how each of the artist is able to “maintain popularity” be due to their consistent release of “notable” songs throughout the years. This is a clear signal on how the artists still maintain their popularity. |

## Popularity: Percentage of un-charted songs released by artist

We have seen the timeline and number of notable songs. Let’s find out how many songs did not chart for each of the artist.

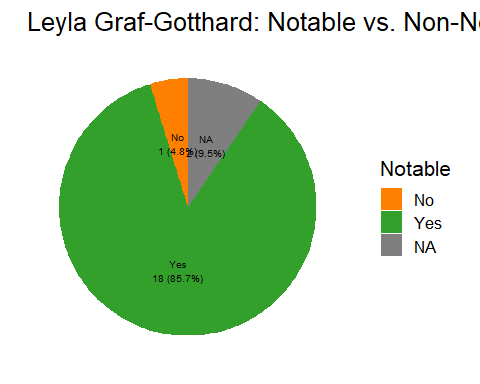
### Kimberly’s songs

# 1. Load & index the graph  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>% mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
# 2. Build an edge index  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from), !is.na(to))  
  
# 3. Find Kimberly Snyder’s index  
kim\_idx <- which(nodes\_tbl$name == "Kimberly Snyder")  
  
# 4. Pull her ComposerOf/PerformerOf song edges, join the Song nodes  
kim\_songs <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"),  
 from == kim\_idx) %>%  
 left\_join(  
 nodes\_tbl %>%   
 filter(`Node Type` == "Song") %>%   
 select(idx, notable),  
 by = c("to" = "idx")  
 )  
  
# 5. Count notable vs non-notable  
summary\_tbl <- kim\_songs %>%  
 mutate(Notable = if\_else(notable, "Yes", "No")) %>%  
 count(Notable) %>%  
 arrange(Notable)  
  
# 6. Plot as a bar chart  
# 6. Plot as a pie chart with labels  
summary\_tbl <- summary\_tbl %>%  
 mutate(percentage = round(n / sum(n) \* 100, 1),  
 label = paste0(Notable, "\n", n, " (", percentage, "%)"))  
  
ggplot(summary\_tbl, aes(x = "", y = n, fill = Notable)) +  
 geom\_bar(stat = "identity", width = 1) +  
 coord\_polar("y", start = 0) +  
 scale\_fill\_manual(values = c(Yes = "#33a02c", No = "#ff7f00")) +  
 geom\_text(aes(label = label), position = position\_stack(vjust = 0.5), size = 3) +  
 labs(  
 title = "Kimberly Snyder: Notable vs. Non-Notable Songs",  
 x = NULL,  
 y = NULL  
 ) +  
 theme\_minimal(base\_size = 16) +  
 theme(axis.text.x = element\_blank(),   
 axis.ticks = element\_blank(),   
 panel.grid = element\_blank())



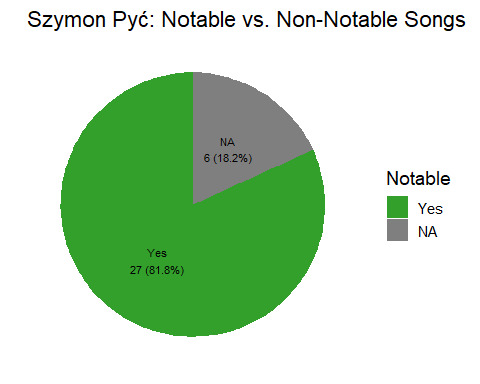
### Leyla’s songs

# 3. Find Kimberly Snyder’s index  
leyla\_idx <- which(nodes\_tbl$name == "Leyla Graf-Gotthard")  
  
# 4. Pull her ComposerOf/PerformerOf song edges, join the Song nodes  
leyla\_songs <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"),  
 from == leyla\_idx) %>%  
 left\_join(  
 nodes\_tbl %>%   
 filter(`Node Type` == "Song") %>%   
 select(idx, notable),  
 by = c("to" = "idx")  
 )  
  
# 5. Count notable vs non-notable  
summary\_tbl2 <- leyla\_songs %>%  
 mutate(Notable = if\_else(notable, "Yes", "No")) %>%  
 count(Notable) %>%  
 arrange(Notable)  
  
# 6. Plot as a bar chart  
# 6. Plot as a pie chart with labels  
summary\_tbl2 <- summary\_tbl2 %>%  
 mutate(percentage = round(n / sum(n) \* 100, 1),  
 label = paste0(Notable, "\n", n, " (", percentage, "%)"))  
  
ggplot(summary\_tbl2, aes(x = "", y = n, fill = Notable)) +  
 geom\_bar(stat = "identity", width = 1) +  
 coord\_polar("y", start = 0) +  
 scale\_fill\_manual(values = c(Yes = "#33a02c", No = "#ff7f00")) +  
 geom\_text(aes(label = label), position = position\_stack(vjust = 0.5), size = 2.6) +  
 labs(  
 title = "Leyla Graf-Gotthard: Notable vs. Non-Notable Songs",  
 x = NULL,  
 y = NULL  
 ) +  
 theme\_minimal(base\_size = 16) +  
 theme(axis.text.x = element\_blank(),   
 axis.ticks = element\_blank(),   
 panel.grid = element\_blank())



### Szymon’s songs

# 3. Find Kimberly Snyder’s index  
szymon\_idx <- which(nodes\_tbl$name == "Szymon Pyć")  
  
# 4. Pull her ComposerOf/PerformerOf song edges, join the Song nodes  
szymon\_songs <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf"),  
 from == szymon\_idx) %>%  
 left\_join(  
 nodes\_tbl %>%   
 filter(`Node Type` == "Song") %>%   
 select(idx, notable),  
 by = c("to" = "idx")  
 )  
  
# 5. Count notable vs non-notable  
summary\_tbl3 <- szymon\_songs %>%  
 mutate(Notable = if\_else(notable, "Yes", "No")) %>%  
 count(Notable) %>%  
 arrange(Notable)  
  
# 6. Plot as a bar chart  
# 6. Plot as a pie chart with labels  
summary\_tbl3 <- summary\_tbl3 %>%  
 mutate(percentage = round(n / sum(n) \* 100, 1),  
 label = paste0(Notable, "\n", n, " (", percentage, "%)"))  
  
ggplot(summary\_tbl3, aes(x = "", y = n, fill = Notable)) +  
 geom\_bar(stat = "identity", width = 1) +  
 coord\_polar("y", start = 0) +  
 scale\_fill\_manual(values = c(Yes = "#33a02c", No = "#ff7f00")) +  
 geom\_text(aes(label = label), position = position\_stack(vjust = 0.5), size = 3) +  
 labs(  
 title = "Szymon Pyć: Notable vs. Non-Notable Songs",  
 x = NULL,  
 y = NULL  
 ) +  
 theme\_minimal(base\_size = 14) +  
 theme(axis.text.x = element\_blank(),   
 axis.ticks = element\_blank(),   
 panel.grid = element\_blank())



|  |
| --- |
| Notable Songs vs Non-Notable Songs |
| These 3 artist have a majority (more than 2/3) of notable songs as compared to non-notable songs they produced. This is another good signal that shows how the artist is able to maintain popularity by releasing top charting songs consistently. |

## Influence: Work Interactions

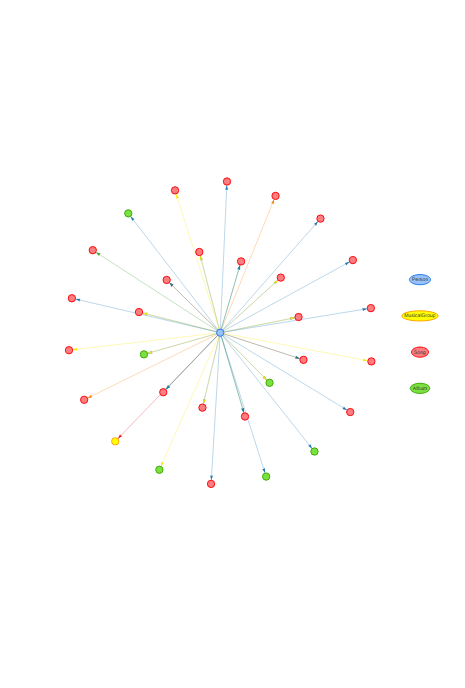
* I want to find out the direct influences by the 3 artist. I will visualise using a visNetwork and count number of **work-type influence edges** each artists has.

|  |
| --- |
| Work-type Influences used in plots |
| Below are the definition of the Edge Types being used.   1. “MemberOf” - Indicates that the source node (Person) is (or was) a member of the destination node (MusicalGroup) 2. “ComposerOf” - Indicates that the source node (Person) composed the destination node (Song or Album) 3. “PerformerOf” - Indicates a that the source node (Person or MusicalGroup) performed the destination node (Song or Album) 4. “LyricistOf” - Indicates that the source node (Person) wrote lyrics for the destination node (Song or Album) 5. “RecordedBy” - Indicates that the destination node (RecordLabel) aided in the recording process for the source node (Song or Album) 6. “ProducerOf” - Indicates that the source node (Person or RecordLabel) participated in the production of the destination node’s work (Song, Album, Person, or MusicalGroup) 7. “DistributedBy” - Indicates that the destination node (RecordLabel) aided in the distribution process for the source node (Song or Album) |

### Kimberly Snyder

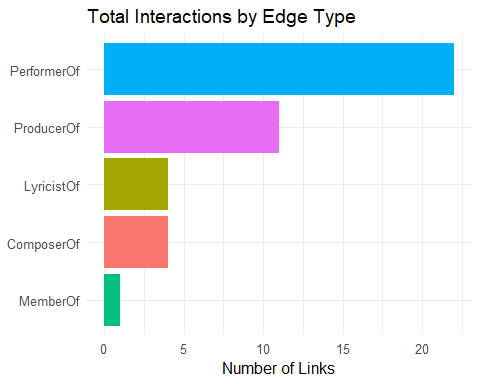
#### Work Influence Edge Types

# 1.  
id\_map <- nodes\_tbl %>%   
 mutate(index = row\_number()) %>%   
 select(id, index)  
  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = index) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = index) %>%  
 filter(!is.na(from) & !is.na(to))  
  
# 2. Filter for your chosen edge types & artist  
artists <- c("Kimberly Snyder")  
edge\_types <- c("MemberOf", "ComposerOf", "PerformerOf","LyricistOf", "RecordedBy","ProducerOf","DistributedBy")  
career\_edges <- edges\_idx %>%  
 filter(`Edge Type` %in% edge\_types,  
 (from %in% which(nodes\_tbl$name %in% artists)) |  
 (to %in% which(nodes\_tbl$name %in% artists)))  
  
# 3. Prune to only the nodes in those edges  
career\_node\_ids <- unique(c(career\_edges$from, career\_edges$to))  
career\_nodes <- nodes\_tbl %>% slice(career\_node\_ids)  
  
# 4. Build visNetwork node DF  
vn\_nodes <- career\_nodes %>%  
 mutate(  
 id = row\_number(),  
 label = name,  
 group = `Node Type`,  
 title = paste0("<b>", name, "</b><br>Type: ", `Node Type`)  
 )  
  
# 5. Build visNetwork edge DF with both label & title  
id\_lu <- tibble(old = career\_node\_ids, new = vn\_nodes$id)  
vn\_edges <- career\_edges %>%  
 inner\_join(id\_lu, by = c("from" = "old")) %>% select(-from) %>% rename(from = new) %>%  
 inner\_join(id\_lu, by = c("to" = "old")) %>% select(-to) %>% rename(to = new) %>%  
 transmute(  
 from,  
 to,  
 label = `Edge Type`, # drawn on the arrow  
 title = `Edge Type`, # hover‐tooltip  
 color = case\_when(  
 label == "MemberOf" ~ "#e31a1c",  
 label == "ComposerOf" ~ "#33a02c",  
 label == "PerformerOf"~ "#1f78b4",  
 label == "LyricistOf" ~ "#ff7f00",  
 label == "RecordedBy" ~ "#6a329f",  
 label == "ProducerOf" ~ "#fce80a",  
 label == "DistributedBy" ~ "#04f3f0",  
 TRUE ~ "#888888"  
 )  
 )  
  
# 6. Plot with edge‐labels  
visNetwork(vn\_nodes, vn\_edges, height = "600px", width = "100%") %>%  
 visNodes(shape = "dot", size =14, font = list(color = "black", size = 10)) %>%  
 visEdges(  
 arrows = "to",  
 labelHighlightBold = TRUE,  
 font = list(color = "blue", size = 8)  
 ) %>%  
 visLegend(useGroups = TRUE, width = 0.1, position = "right") %>%  
 visIgraphLayout(layout = "layout\_with\_fr", randomSeed = 1234) %>%  
 visPhysics(enabled = FALSE)



#### Summary of the Edge interactions

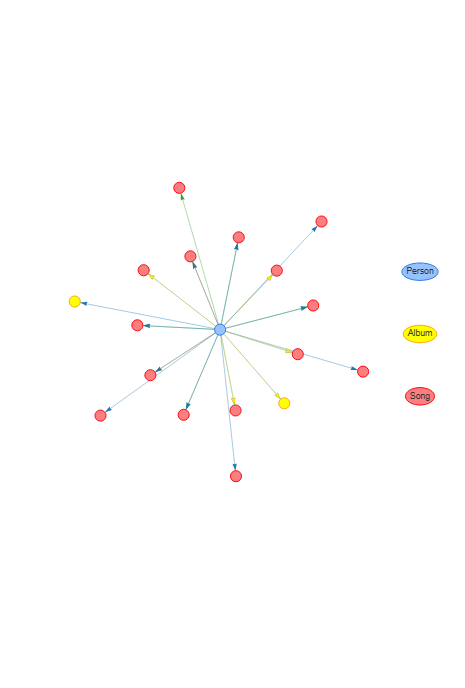
# Assuming `career\_edges` is already in your environment:  
# Count interactions per edge type  
edge\_summary <- career\_edges %>%  
 count(`Edge Type`, name = "Interactions") %>%  
 arrange(Interactions)  
  
# Bar‐chart of interactions by edge type  
ggplot(edge\_summary, aes(  
 x = fct\_reorder(`Edge Type`, Interactions),  
 y = Interactions,  
 fill = `Edge Type`  
 )) +  
 geom\_col(show.legend = FALSE) +  
 coord\_flip() +  
 labs(  
 title = "Total Interactions by Edge Type",  
 x = NULL,  
 y = "Number of Links"  
 ) +  
 theme\_minimal(base\_size = 12)



### Leyla Graf-Gotthard

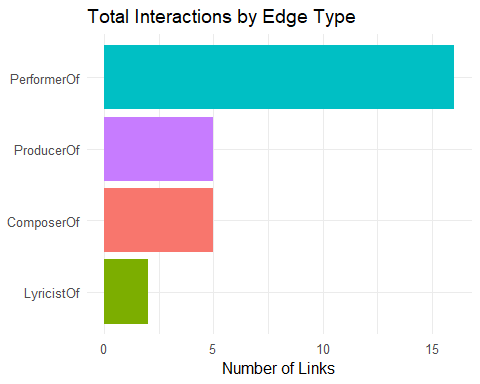
#### Work Influence Edge Types

# 2. Filter for your chosen edge types & artist  
artists2 <- c("Leyla Graf-Gotthard")  
edge\_types <- c("MemberOf", "ComposerOf", "PerformerOf","LyricistOf", "RecordedBy","ProducerOf","DistributedBy")  
career\_edges2 <- edges\_idx %>%  
 filter(`Edge Type` %in% edge\_types,  
 (from %in% which(nodes\_tbl$name %in% artists2)) |  
 (to %in% which(nodes\_tbl$name %in% artists2)))  
  
# 3. Prune to only the nodes in those edges  
career\_node\_ids2 <- unique(c(career\_edges2$from, career\_edges2$to))  
career\_nodes2 <- nodes\_tbl %>% slice(career\_node\_ids2)  
  
# 4. Build visNetwork node DF  
vn\_nodes2 <- career\_nodes2 %>%  
 mutate(  
 id = row\_number(),  
 label = name,  
 group = `Node Type`,  
 title = paste0("<b>", name, "</b><br>Type: ", `Node Type`)  
 )  
  
# 5. Build visNetwork edge DF with both label & title  
id\_lu2 <- tibble(old = career\_node\_ids2, new = vn\_nodes2$id)  
vn\_edges2 <- career\_edges2 %>%  
 inner\_join(id\_lu2, by = c("from" = "old")) %>% select(-from) %>% rename(from = new) %>%  
 inner\_join(id\_lu2, by = c("to" = "old")) %>% select(-to) %>% rename(to = new) %>%  
 transmute(  
 from,  
 to,  
 label = `Edge Type`, # drawn on the arrow  
 title = `Edge Type`, # hover‐tooltip  
 color = case\_when(  
 label == "MemberOf" ~ "#e31a1c",  
 label == "ComposerOf" ~ "#33a02c",  
 label == "PerformerOf"~ "#1f78b4",  
 label == "LyricistOf" ~ "#ff7f00",  
 label == "RecordedBy" ~ "#6a329f",  
 label == "ProducerOf" ~ "#fce80a",  
 label == "DistributedBy" ~ "#04f3f0",  
 TRUE ~ "#888888"  
 )  
 )  
  
# 6. Plot with edge‐labels  
visNetwork(vn\_nodes2, vn\_edges2, height = "600px", width = "100%") %>%  
 visNodes(shape = "dot", size =14, font = list(color = "black", size = 10)) %>%  
 visEdges(  
 arrows = "to",  
 labelHighlightBold = TRUE,  
 font = list(color = "blue", size = 8)  
 ) %>%  
 visLegend(useGroups = TRUE, width = 0.1, position = "right") %>%  
 visIgraphLayout(layout = "layout\_with\_fr", randomSeed = 1234) %>%  
 visPhysics(enabled = FALSE)



#### Summary of the interactions

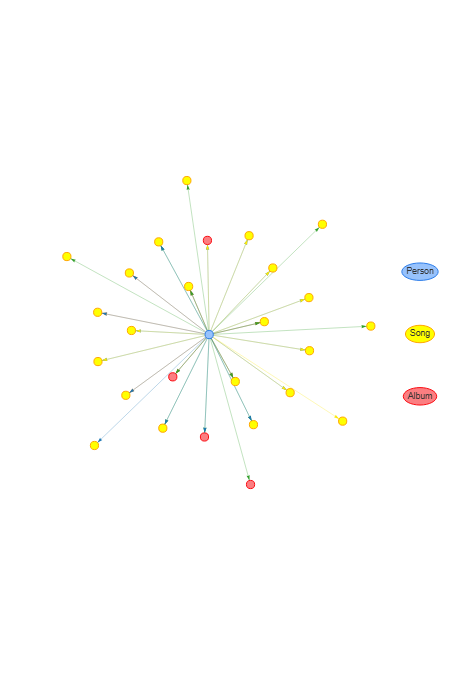
# Assuming `career\_edges` is already in your environment:  
# Count interactions per edge type  
edge\_summary2 <- career\_edges2 %>%  
 count(`Edge Type`, name = "Interactions") %>%  
 arrange(Interactions)  
  
# Bar‐chart of interactions by edge type  
ggplot(edge\_summary2, aes(  
 x = fct\_reorder(`Edge Type`, Interactions),  
 y = Interactions,  
 fill = `Edge Type`  
 )) +  
 geom\_col(show.legend = FALSE) +  
 coord\_flip() +  
 labs(  
 title = "Total Interactions by Edge Type",  
 x = NULL,  
 y = "Number of Links"  
 ) +  
 theme\_minimal(base\_size = 12)



### Szymon Pyć

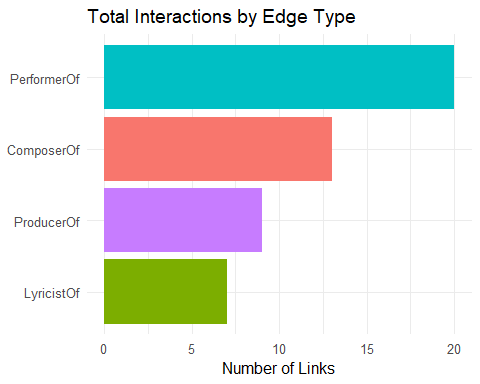
#### Work Influence Edge Types

# 2. Filter for your chosen edge types & artist  
artists3 <- c("Szymon Pyć")  
edge\_types <- c("MemberOf", "ComposerOf", "PerformerOf","LyricistOf", "RecordedBy","ProducerOf","DistributedBy")  
career\_edges3 <- edges\_idx %>%  
 filter(`Edge Type` %in% edge\_types,  
 (from %in% which(nodes\_tbl$name %in% artists3)) |  
 (to %in% which(nodes\_tbl$name %in% artists3)))  
  
# 3. Prune to only the nodes in those edges  
career\_node\_ids3 <- unique(c(career\_edges3$from, career\_edges3$to))  
career\_nodes3 <- nodes\_tbl %>% slice(career\_node\_ids3)  
  
# 4. Build visNetwork node DF  
vn\_nodes3 <- career\_nodes3 %>%  
 mutate(  
 id = row\_number(),  
 label = name,  
 group = `Node Type`,  
 title = paste0("<b>", name, "</b><br>Type: ", `Node Type`)  
 )  
  
# 5. Build visNetwork edge DF with both label & title  
id\_lu3 <- tibble(old = career\_node\_ids3, new = vn\_nodes3$id)  
vn\_edges3 <- career\_edges3 %>%  
 inner\_join(id\_lu3, by = c("from" = "old")) %>% select(-from) %>% rename(from = new) %>%  
 inner\_join(id\_lu3, by = c("to" = "old")) %>% select(-to) %>% rename(to = new) %>%  
 transmute(  
 from,  
 to,  
 label = `Edge Type`, # drawn on the arrow  
 title = `Edge Type`, # hover‐tooltip  
 color = case\_when(  
 label == "MemberOf" ~ "#e31a1c",  
 label == "ComposerOf" ~ "#33a02c",  
 label == "PerformerOf"~ "#1f78b4",  
 label == "LyricistOf" ~ "#ff7f00",  
 label == "RecordedBy" ~ "#6a329f",  
 label == "ProducerOf" ~ "#fce80a",  
 label == "DistributedBy" ~ "#04f3f0",  
 TRUE ~ "#888888"  
 )  
 )  
  
# 6. Plot with edge‐labels  
visNetwork(vn\_nodes3, vn\_edges3, height = "600px", width = "100%") %>%  
 visNodes(shape = "dot", size =14, font = list(color = "black", size = 10)) %>%  
 visEdges(  
 arrows = "to",  
 labelHighlightBold = TRUE,  
 font = list(color = "blue", size = 8)  
 ) %>%  
 visLegend(useGroups = TRUE, width = 0.1, position = "right") %>%  
 visIgraphLayout(layout = "layout\_with\_fr", randomSeed = 1234) %>%  
 visPhysics(enabled = FALSE)



#### Summary of the interactions

# Assuming `career\_edges` is already in your environment:  
# Count interactions per edge type  
edge\_summary3 <- career\_edges3 %>%  
 count(`Edge Type`, name = "Interactions") %>%  
 arrange(Interactions)  
  
# Bar‐chart of interactions by edge type  
ggplot(edge\_summary3, aes(  
 x = fct\_reorder(`Edge Type`, Interactions),  
 y = Interactions,  
 fill = `Edge Type`  
 )) +  
 geom\_col(show.legend = FALSE) +  
 coord\_flip() +  
 labs(  
 title = "Total Interactions by Edge Type",  
 x = NULL,  
 y = "Number of Links"  
 ) +  
 theme\_minimal(base\_size = 12)



|  |
| --- |
| Work-type Influences |
| These plots show the number of work influences by the artist within their scope. |

## Influence: People Interactions

This portion will cover how people are directly/indirectly influenced by the selected artists and their work. I will be plotting a visNetwork and barcharts for Betweenness Centrality value to show their influences.

|  |
| --- |
| People Influences |
| 1. “InterpolatesFrom” - Indicates that the source node (Song or Album) interpolated a melody from the destination node (Song or Album). 2. “DirectlySamples” - Indicates that the source node (Song or Album) consists of (an) audio recording(s) that directly reuse a portion of the audio recording of the destination node (Song or Album) via sampling. 3. “InStyleOf” - Indicates that the source node (Song or Album) consists of (an) audio recording(s) that directly reuse a portion of the audio recording of the destination node (Song or Album) via sampling. 4. “LyricalReferenceTo” - Indicates that the source node (Song or Album) consists of (an) audio recording(s) that directly reuse a portion of the audio recording of the destination node (Song or Album) via sampling. 5. “CoverOf” - Indicates that the source node (Song or Album) is a cover of the destination node (Song or Album). |

**VisNetwork**

* The visNetwork will be limited to a maximum hop of 2 to reduce messiness.
* Dropdown select will be created to see the individual artists and their influences.

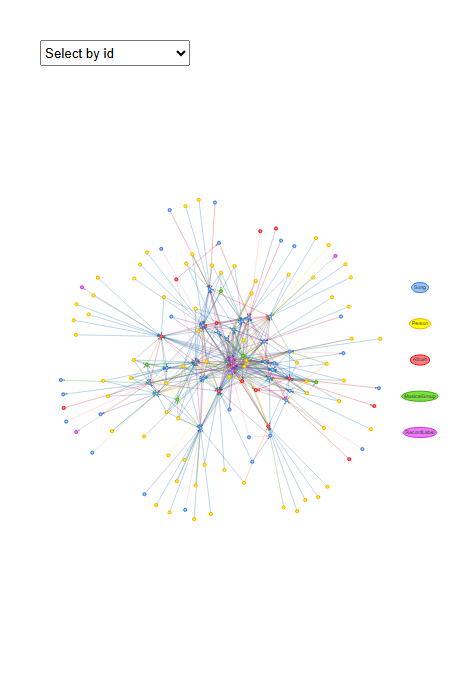
**Betweenness Centrality Value**

* This measures a node’s influence in a network by quantifying how often it lies on the shortest paths between other nodes.
* I will solely using this as information on what are the most relevant “bridges” controlling the flow of in the network using maximum hop of 3.

### Kimberly Snyder

#### People Influence Edges

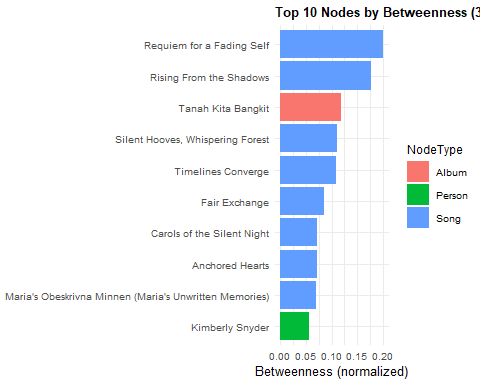
# ─── 0. How many hops? ────────────────────────────────────────────────────────  
max\_hops <- 2 # Change this number to 1, 2, 3… then re‐run the chunk  
  
# ─── 1. Load & index all nodes and edges ────────────────────────────────────  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>% mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
# Build numeric “from/to” index for every edge  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from) & !is.na(to))  
  
# ─── 2. Build an undirected igraph of the full graph ────────────────────────  
g\_all <- graph\_from\_data\_frame(  
 d = edges\_idx %>% select(from, to),  
 directed = FALSE,  
 vertices = nodes\_tbl %>% transmute(name = as.character(idx))  
)  
  
# ─── 3. Prepare list of all Person names for dropdown ───────────────────────  
all\_persons <- nodes\_tbl %>%  
 filter(`Node Type` == "Person") %>%  
 arrange(name) %>% # sort ascending by name  
 pull(name)  
  
# ─── 4. Create a named vector mapping Person ➔ idx (character) ─────────────  
person\_idx\_map <- nodes\_tbl %>%  
 filter(`Node Type` == "Person") %>%  
 transmute(Person = name, idx\_char = as.character(idx)) %>%  
 deframe()  
  
# ─── 5. Function to build and render a Kim‐style ego network given a name ───  
render\_ego\_network <- function(center\_name) {  
 center\_idx <- person\_idx\_map[[center\_name]]  
 if (is.null(center\_idx)) {  
 stop("Person '", center\_name, "' not found.")  
 }  
 # Compute ego‐set of all vertices within max\_hops of chosen person  
 ego\_list <- ego(g\_all, order = max\_hops, nodes = center\_idx, mode = "all")[[1]]  
 ego\_indices <- as.integer(V(g\_all)$name[ego\_list])  
 # Filter edges so both endpoints are in that ego set  
 career\_edges <- edges\_idx %>%  
 filter(from %in% ego\_indices, to %in% ego\_indices)  
 if (nrow(career\_edges) == 0) {  
 showNotification(paste0(  
 "No edges remain when limiting to ", max\_hops,  
 " hops out from ", center\_name, "."  
 ), type = "warning")  
 return(NULL)  
 }  
 # Build list of all involved nodes  
 career\_node\_ids <- unique(c(career\_edges$from, career\_edges$to))  
 career\_nodes <- nodes\_tbl %>% slice(career\_node\_ids) %>%  
 mutate(  
 vis\_id = row\_number(),  
 label = name,  
 group = `Node Type`,  
 title = paste0("<b>", name, "</b><br>Type: ", `Node Type`,  
 if\_else(`Node Type` == "Song" & !is.na(release\_date),  
 paste0("<br>Released: ", release\_date), ""))  
 )  
 # Remap edges to vis\_ids  
 career\_edges <- career\_edges %>%  
 mutate(  
 from\_vis = match(from, career\_node\_ids),  
 to\_vis = match(to, career\_node\_ids)  
 )  
 # Construct visNetwork nodes & edges  
 vn\_nodes <- career\_nodes %>%  
 transmute(id = vis\_id, label = label, group = group, title = title)  
 vn\_edges <- career\_edges %>%  
 transmute(from = from\_vis, to = to\_vis, label = `Edge Type`, title = `Edge Type`,  
 color = case\_when(  
 label == "InterpolatesFrom" ~ "#1f78b4",  
 label == "DirectlySamples" ~ "#33a02c",  
 label == "InStyleOf" ~ "#e31a1c",  
 label == "LyricalReferenceTo" ~ "#6a3d9a",  
 label == "CoverOf" ~ "#fb9a99",  
 label == "ComposerOf" ~ "#33a02c",  
 label == "PerformerOf" ~ "#1f78b4",  
 label == "RecordedBy" ~ "#6a329f",  
 label == "ProducerOf" ~ "#fce80a",  
 label == "LyricistOf" ~ "#ff7f00",  
 TRUE ~ "#888888"  
 ))  
 # Determine vis\_id for all Person‐type nodes (to allow dropdown focus)  
 person\_vis\_ids <- vn\_nodes %>%  
 filter(group == "Person") %>%  
 arrange(label) %>% # ensure sorted ascending  
 pull(id)  
 # Render visNetwork  
 visNetwork(vn\_nodes, vn\_edges, height = "600px", width = "100%") %>%  
 visNodes(shape = "dot", size = 14, font = list(color = "black", size = 10)) %>%  
 visEdges(arrows = "to", labelHighlightBold = TRUE, font = list(color = "blue", size = 7)) %>%  
 visOptions(  
 highlightNearest = list(enabled = TRUE, degree = 1),  
 nodesIdSelection = list(  
 enabled = TRUE,  
 useLabels = TRUE,  
 values = person\_vis\_ids  
 )  
 ) %>%  
 visLegend(useGroups = TRUE, width = 0.1, position = "right") %>%  
 visIgraphLayout(layout = "layout\_with\_fr", randomSeed = 1234) %>%  
 visPhysics(enabled = FALSE)  
}  
  
# ─── 6. Initial rendering for “Kimberly Snyder” ─────────────────────────────  
render\_ego\_network("Kimberly Snyder")



#### Betweenness centrality in network

* Kimberly had more song influences in her network.

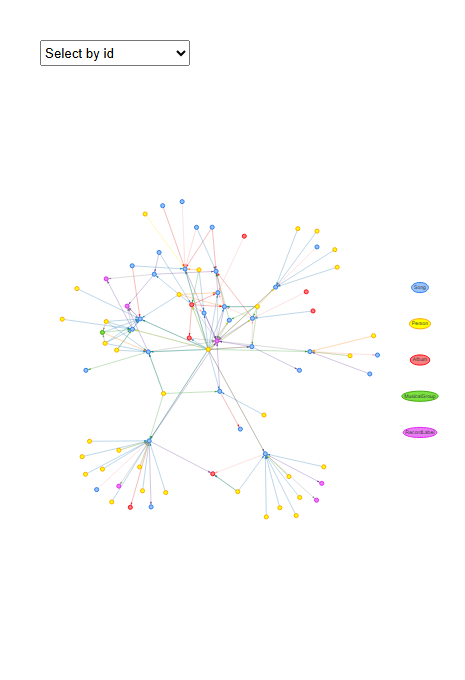
# ─── PARAMETERS ───────────────────────────────────────────────────────────────  
center\_person <- "Kimberly Snyder"  
max\_hops <- 3  
keep\_edge\_types <- c(  
 "InterpolatesFrom", "InStyleOf", "LyricalReferenceTo",  
 "CoverOf", "DirectlySamples",  
 "ComposerOf", "PerformerOf", "MemberOf", "Released"  
)  
  
# ─── 1. Load & index all nodes and edges ────────────────────────────────────  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>% mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from) & !is.na(to))  
  
# ─── 2. Build an undirected igraph of the full graph (topology only) ────────  
g\_all <- graph\_from\_data\_frame(  
 d = edges\_idx %>% select(from, to),  
 directed = FALSE,  
 vertices = nodes\_tbl %>% transmute(name = as.character(idx))  
)  
  
# ─── 3. Find the chosen person’s idx (character) ────────────────────────────  
center\_idx\_char <- nodes\_tbl %>%  
 filter(name == center\_person, `Node Type` == "Person") %>%  
 pull(idx) %>%  
 as.character()  
if (length(center\_idx\_char) != 1) {  
 stop("Person '", center\_person, "' not found or not unique.")  
}  
  
# ─── 4. Compute the ego‐set of vertices within max\_hops of the center ────────  
ego\_list <- ego(g\_all, order = max\_hops, nodes = center\_idx\_char, mode = "all")[[1]]  
ego\_indices <- as.integer(V(g\_all)$name[ego\_list]) # numeric idx of all nodes in ego‐set  
  
# ─── 5. Filter edges so that both endpoints lie in that ego‐set AND edge‐type is in keep\_edge\_types ─  
ego\_edges <- edges\_idx %>%  
 filter(  
 from %in% ego\_indices,  
 to %in% ego\_indices,  
 `Edge Type` %in% keep\_edge\_types  
 )  
  
if (nrow(ego\_edges) == 0) {  
 message("No edges remain when limiting to ", max\_hops, " hop(s) around ", center\_person, ".")  
} else {  
 # ─── 6. Build the list of involved nodes in this ego‐set ─────────────────────  
 ego\_node\_ids <- unique(c(ego\_edges$from, ego\_edges$to))  
 ego\_nodes <- nodes\_tbl %>% slice(ego\_node\_ids)  
   
 # ─── 7. Construct an igraph for this ego network ───────────────────────────  
 vertices\_df <- ego\_nodes %>%  
 transmute(name = as.character(idx), label = name, type = `Node Type`)  
 edges\_df <- ego\_edges %>%  
 transmute(from = as.character(from), to = as.character(to), etype = `Edge Type`)  
   
 g\_ego <- graph\_from\_data\_frame(  
 d = edges\_df %>% select(from, to),  
 directed = FALSE,  
 vertices = vertices\_df  
 )  
   
 # ─── 8. Compute betweenness centrality for every vertex ────────────────────  
 bc\_vals <- betweenness(g\_ego, directed = FALSE, normalized = TRUE)  
 centrality\_tbl <- tibble(  
 idx = as.integer(names(bc\_vals)),  
 betweenness = unname(bc\_vals)  
 ) %>%  
 left\_join(  
 ego\_nodes %>% select(idx, NodeName = name, NodeType = `Node Type`),  
 by = "idx"  
 ) %>%  
 arrange(desc(betweenness))  
   
   
 # ─── 9. Bar‐chart of top 10 nodes by betweenness ──────────────────────────  
 centrality\_tbl %>%  
 slice\_head(n = 10) %>%  
 ggplot(aes(x = fct\_reorder(NodeName, betweenness), y = betweenness, fill = NodeType)) +  
 geom\_col(show.legend = TRUE) +  
 coord\_flip() +  
 labs(  
 title = paste0("Top 10 Nodes by Betweenness (", max\_hops, "-hop around ", center\_person, ")"),  
 x = NULL,  
 y = "Betweenness (normalized)"  
 ) +  
 theme\_minimal(base\_size = 9) +  
 theme(  
 plot.title = element\_text(size = 10, face = "bold"),  
 axis.title.y = element\_text(size = 5),  
 axis.title.x = element\_text(size = 10)  
 )  
}



### Leyla Graf-Gotthard

#### People Influence Edges

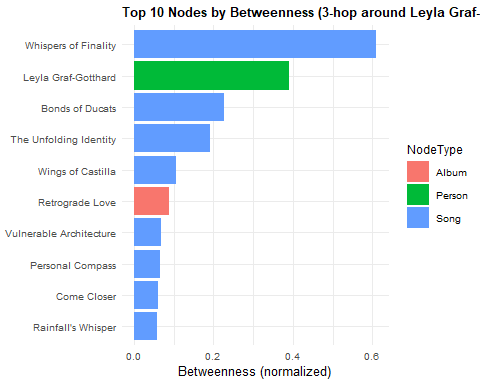
# ─── 0. How many hops? ────────────────────────────────────────────────────────  
max\_hops <- 2 # Change this number to 1, 2, 3… then re‐run the chunk  
  
# ─── 1. Load & index all nodes and edges ────────────────────────────────────  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>% mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
# Build numeric “from/to” index for every edge  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from) & !is.na(to))  
  
# ─── 2. Build an undirected igraph of the full graph ────────────────────────  
g\_all <- graph\_from\_data\_frame(  
 d = edges\_idx %>% select(from, to),  
 directed = FALSE,  
 vertices = nodes\_tbl %>% transmute(name = as.character(idx))  
)  
  
# ─── 3. Prepare list of all Person names for dropdown ───────────────────────  
all\_persons <- nodes\_tbl %>%  
 filter(`Node Type` == "Person") %>%  
 arrange(name) %>% # sort ascending by name  
 pull(name)  
  
# ─── 4. Create a named vector mapping Person ➔ idx (character) ─────────────  
person\_idx\_map <- nodes\_tbl %>%  
 filter(`Node Type` == "Person") %>%  
 transmute(Person = name, idx\_char = as.character(idx)) %>%  
 deframe()  
  
# ─── 5. Function to build and render a Leyla‐style ego network given a name ───  
render\_ego\_network <- function(center\_name) {  
 center\_idx <- person\_idx\_map[[center\_name]]  
 if (is.null(center\_idx)) {  
 stop("Person '", center\_name, "' not found.")  
 }  
 # Compute ego‐set of all vertices within max\_hops of chosen person  
 ego\_list <- ego(g\_all, order = max\_hops, nodes = center\_idx, mode = "all")[[1]]  
 ego\_indices <- as.integer(V(g\_all)$name[ego\_list])  
 # Filter edges so both endpoints are in that ego set  
 career\_edges <- edges\_idx %>%  
 filter(from %in% ego\_indices, to %in% ego\_indices)  
 if (nrow(career\_edges) == 0) {  
 showNotification(paste0(  
 "No edges remain when limiting to ", max\_hops,  
 " hops out from ", center\_name, "."  
 ), type = "warning")  
 return(NULL)  
 }  
 # Build list of all involved nodes  
 career\_node\_ids <- unique(c(career\_edges$from, career\_edges$to))  
 career\_nodes <- nodes\_tbl %>% slice(career\_node\_ids) %>%  
 mutate(  
 vis\_id = row\_number(),  
 label = name,  
 group = `Node Type`,  
 title = paste0("<b>", name, "</b><br>Type: ", `Node Type`,  
 if\_else(`Node Type` == "Song" & !is.na(release\_date),  
 paste0("<br>Released: ", release\_date), ""))  
 )  
 # Remap edges to vis\_ids  
 career\_edges <- career\_edges %>%  
 mutate(  
 from\_vis = match(from, career\_node\_ids),  
 to\_vis = match(to, career\_node\_ids)  
 )  
 # Construct visNetwork nodes & edges  
 vn\_nodes <- career\_nodes %>%  
 transmute(id = vis\_id, label = label, group = group, title = title)  
 vn\_edges <- career\_edges %>%  
 transmute(from = from\_vis, to = to\_vis, label = `Edge Type`, title = `Edge Type`,  
 color = case\_when(  
 label == "InterpolatesFrom" ~ "#1f78b4",  
 label == "DirectlySamples" ~ "#33a02c",  
 label == "InStyleOf" ~ "#e31a1c",  
 label == "LyricalReferenceTo" ~ "#6a3d9a",  
 label == "CoverOf" ~ "#fb9a99",  
 label == "ComposerOf" ~ "#33a02c",  
 label == "PerformerOf" ~ "#1f78b4",  
 label == "RecordedBy" ~ "#6a329f",  
 label == "ProducerOf" ~ "#fce80a",  
 label == "LyricistOf" ~ "#ff7f00",  
 TRUE ~ "#888888"  
 ))  
 # Determine vis\_id for all Person‐type nodes (to allow dropdown focus)  
 person\_vis\_ids <- vn\_nodes %>%  
 filter(group == "Person") %>%  
 arrange(label) %>% # ensure sorted ascending  
 pull(id)  
 # Render visNetwork  
 visNetwork(vn\_nodes, vn\_edges, height = "600px", width = "100%") %>%  
 visNodes(shape = "dot", size = 14, font = list(color = "black", size = 10)) %>%  
 visEdges(arrows = "to", labelHighlightBold = TRUE, font = list(color = "blue", size = 7)) %>%  
 visOptions(  
 highlightNearest = list(enabled = TRUE, degree = 1),  
 nodesIdSelection = list(  
 enabled = TRUE,  
 useLabels = TRUE,  
 values = person\_vis\_ids  
 )  
 ) %>%  
 visLegend(useGroups = TRUE, width = 0.1, position = "right") %>%  
 visIgraphLayout(layout = "layout\_with\_fr", randomSeed = 1234) %>%  
 visPhysics(enabled = FALSE)  
}  
  
# ─── 6. Initial rendering for “Kimberly Snyder” ─────────────────────────────  
render\_ego\_network("Leyla Graf-Gotthard")



#### Betweenness centrality in network

* Similarly, Leyla had more song influences in her network.

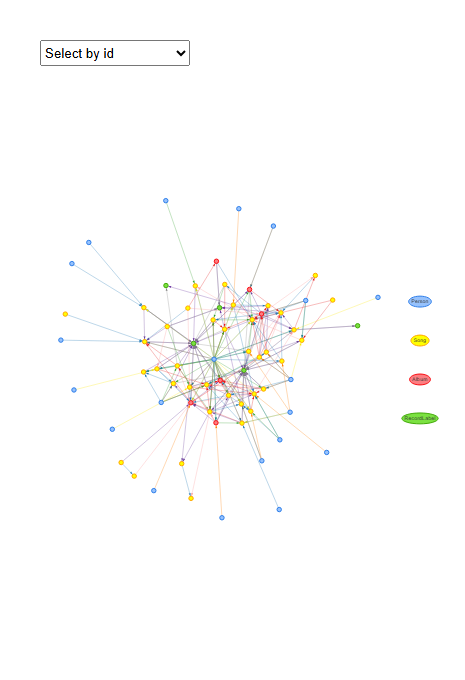
# ─── PARAMETERS ───────────────────────────────────────────────────────────────  
center\_person <- "Leyla Graf-Gotthard"  
max\_hops <- 3  
keep\_edge\_types <- c(  
 "InterpolatesFrom", "InStyleOf", "LyricalReferenceTo",  
 "CoverOf", "DirectlySamples",  
 "ComposerOf", "PerformerOf", "MemberOf", "Released"  
)  
  
# ─── 1. Load & index all nodes and edges ────────────────────────────────────  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>% mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from) & !is.na(to))  
  
# ─── 2. Build an undirected igraph of the full graph (topology only) ────────  
g\_all <- graph\_from\_data\_frame(  
 d = edges\_idx %>% select(from, to),  
 directed = FALSE,  
 vertices = nodes\_tbl %>% transmute(name = as.character(idx))  
)  
  
# ─── 3. Find the chosen person’s idx (character) ────────────────────────────  
center\_idx\_char <- nodes\_tbl %>%  
 filter(name == center\_person, `Node Type` == "Person") %>%  
 pull(idx) %>%  
 as.character()  
if (length(center\_idx\_char) != 1) {  
 stop("Person '", center\_person, "' not found or not unique.")  
}  
  
# ─── 4. Compute the ego‐set of vertices within max\_hops of the center ────────  
ego\_list <- ego(g\_all, order = max\_hops, nodes = center\_idx\_char, mode = "all")[[1]]  
ego\_indices <- as.integer(V(g\_all)$name[ego\_list]) # numeric idx of all nodes in ego‐set  
  
# ─── 5. Filter edges so that both endpoints lie in that ego‐set AND edge‐type is in keep\_edge\_types ─  
ego\_edges <- edges\_idx %>%  
 filter(  
 from %in% ego\_indices,  
 to %in% ego\_indices,  
 `Edge Type` %in% keep\_edge\_types  
 )  
  
if (nrow(ego\_edges) == 0) {  
 message("No edges remain when limiting to ", max\_hops, " hop(s) around ", center\_person, ".")  
} else {  
 # ─── 6. Build the list of involved nodes in this ego‐set ─────────────────────  
 ego\_node\_ids <- unique(c(ego\_edges$from, ego\_edges$to))  
 ego\_nodes <- nodes\_tbl %>% slice(ego\_node\_ids)  
   
 # ─── 7. Construct an igraph for this ego network ───────────────────────────  
 vertices\_df <- ego\_nodes %>%  
 transmute(name = as.character(idx), label = name, type = `Node Type`)  
 edges\_df <- ego\_edges %>%  
 transmute(from = as.character(from), to = as.character(to), etype = `Edge Type`)  
   
 g\_ego <- graph\_from\_data\_frame(  
 d = edges\_df %>% select(from, to),  
 directed = FALSE,  
 vertices = vertices\_df  
 )  
   
 # ─── 8. Compute betweenness centrality for every vertex ────────────────────  
 bc\_vals <- betweenness(g\_ego, directed = FALSE, normalized = TRUE)  
 centrality\_tbl <- tibble(  
 idx = as.integer(names(bc\_vals)),  
 betweenness = unname(bc\_vals)  
 ) %>%  
 left\_join(  
 ego\_nodes %>% select(idx, NodeName = name, NodeType = `Node Type`),  
 by = "idx"  
 ) %>%  
 arrange(desc(betweenness))  
   
 # ─── 9. Bar‐chart of top 10 nodes by betweenness ──────────────────────────  
 centrality\_tbl %>%  
 slice\_head(n = 10) %>%  
 ggplot(aes(x = fct\_reorder(NodeName, betweenness), y = betweenness, fill = NodeType)) +  
 geom\_col(show.legend = TRUE) +  
 coord\_flip() +  
 labs(  
 title = paste0("Top 10 Nodes by Betweenness (", max\_hops, "-hop around ", center\_person, ")"),  
 x = NULL,  
 y = "Betweenness (normalized)"  
 ) +  
 theme\_minimal(base\_size = 9) +  
 theme(  
 plot.title = element\_text(size = 10, face = "bold"),  
 axis.title.y = element\_text(size = 5),  
 axis.title.x = element\_text(size = 10)  
 )  
}



### Szymon Pyć

#### People Influence Edges

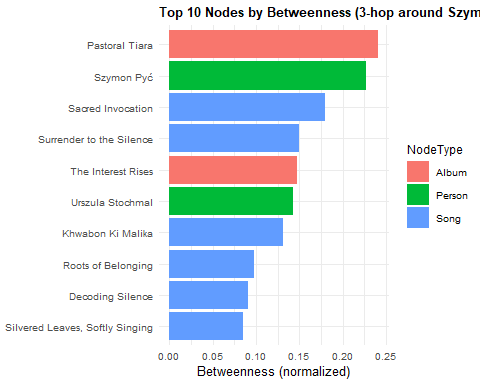
# ─── 0. How many hops? ────────────────────────────────────────────────────────  
max\_hops <- 2 # Change this number to 1, 2, 3… then re‐run the chunk  
  
# ─── 1. Load & index all nodes and edges ────────────────────────────────────  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>% mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
# Build numeric “from/to” index for every edge  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from) & !is.na(to))  
  
# ─── 2. Build an undirected igraph of the full graph ────────────────────────  
g\_all <- graph\_from\_data\_frame(  
 d = edges\_idx %>% select(from, to),  
 directed = FALSE,  
 vertices = nodes\_tbl %>% transmute(name = as.character(idx))  
)  
  
# ─── 3. Prepare list of all Person names for dropdown ───────────────────────  
all\_persons <- nodes\_tbl %>%  
 filter(`Node Type` == "Person") %>%  
 arrange(name) %>% # sort ascending by name  
 pull(name)  
  
# ─── 4. Create a named vector mapping Person ➔ idx (character) ─────────────  
person\_idx\_map <- nodes\_tbl %>%  
 filter(`Node Type` == "Person") %>%  
 transmute(Person = name, idx\_char = as.character(idx)) %>%  
 deframe()  
  
# ─── 5. Function to build and render a Leyla‐style ego network given a name ───  
render\_ego\_network <- function(center\_name) {  
 center\_idx <- person\_idx\_map[[center\_name]]  
 if (is.null(center\_idx)) {  
 stop("Person '", center\_name, "' not found.")  
 }  
 # Compute ego‐set of all vertices within max\_hops of chosen person  
 ego\_list <- ego(g\_all, order = max\_hops, nodes = center\_idx, mode = "all")[[1]]  
 ego\_indices <- as.integer(V(g\_all)$name[ego\_list])  
 # Filter edges so both endpoints are in that ego set  
 career\_edges <- edges\_idx %>%  
 filter(from %in% ego\_indices, to %in% ego\_indices)  
 if (nrow(career\_edges) == 0) {  
 showNotification(paste0(  
 "No edges remain when limiting to ", max\_hops,  
 " hops out from ", center\_name, "."  
 ), type = "warning")  
 return(NULL)  
 }  
 # Build list of all involved nodes  
 career\_node\_ids <- unique(c(career\_edges$from, career\_edges$to))  
 career\_nodes <- nodes\_tbl %>% slice(career\_node\_ids) %>%  
 mutate(  
 vis\_id = row\_number(),  
 label = name,  
 group = `Node Type`,  
 title = paste0("<b>", name, "</b><br>Type: ", `Node Type`,  
 if\_else(`Node Type` == "Song" & !is.na(release\_date),  
 paste0("<br>Released: ", release\_date), ""))  
 )  
 # Remap edges to vis\_ids  
 career\_edges <- career\_edges %>%  
 mutate(  
 from\_vis = match(from, career\_node\_ids),  
 to\_vis = match(to, career\_node\_ids)  
 )  
 # Construct visNetwork nodes & edges  
 vn\_nodes <- career\_nodes %>%  
 transmute(id = vis\_id, label = label, group = group, title = title)  
 vn\_edges <- career\_edges %>%  
 transmute(from = from\_vis, to = to\_vis, label = `Edge Type`, title = `Edge Type`,  
 color = case\_when(  
 label == "InterpolatesFrom" ~ "#1f78b4",  
 label == "DirectlySamples" ~ "#33a02c",  
 label == "InStyleOf" ~ "#e31a1c",  
 label == "LyricalReferenceTo" ~ "#6a3d9a",  
 label == "CoverOf" ~ "#fb9a99",  
 label == "ComposerOf" ~ "#33a02c",  
 label == "PerformerOf" ~ "#1f78b4",  
 label == "RecordedBy" ~ "#6a329f",  
 label == "ProducerOf" ~ "#fce80a",  
 label == "LyricistOf" ~ "#ff7f00",  
 TRUE ~ "#888888"  
 ))  
 # Determine vis\_id for all Person‐type nodes (to allow dropdown focus)  
 person\_vis\_ids <- vn\_nodes %>%  
 filter(group == "Person") %>%  
 arrange(label) %>% # ensure sorted ascending  
 pull(id)  
 # Render visNetwork  
 visNetwork(vn\_nodes, vn\_edges, height = "600px", width = "100%") %>%  
 visNodes(shape = "dot", size = 14, font = list(color = "black", size = 10)) %>%  
 visEdges(arrows = "to", labelHighlightBold = TRUE, font = list(color = "blue", size = 7)) %>%  
 visOptions(  
 highlightNearest = list(enabled = TRUE, degree = 1),  
 nodesIdSelection = list(  
 enabled = TRUE,  
 useLabels = TRUE,  
 values = person\_vis\_ids  
 )  
 ) %>%  
 visLegend(useGroups = TRUE, width = 0.1, position = "right") %>%  
 visIgraphLayout(layout = "layout\_with\_fr", randomSeed = 1234) %>%  
 visPhysics(enabled = FALSE)  
}  
  
# ─── 6. Initial rendering for “Kimberly Snyder” ─────────────────────────────  
render\_ego\_network("Szymon Pyć")



#### Betweenness centrality in network

* Interestingly, there were more albums and people influence in Szymon’s network.

# ─── PARAMETERS ───────────────────────────────────────────────────────────────  
center\_person <- "Szymon Pyć"  
max\_hops <- 3  
keep\_edge\_types <- c(  
 "InterpolatesFrom", "InStyleOf", "LyricalReferenceTo",  
 "CoverOf", "DirectlySamples",  
 "ComposerOf", "PerformerOf", "MemberOf", "Released"  
)  
  
# ─── 1. Load & index all nodes and edges ────────────────────────────────────  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>% mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from) & !is.na(to))  
  
# ─── 2. Build an undirected igraph of the full graph (topology only) ────────  
g\_all <- graph\_from\_data\_frame(  
 d = edges\_idx %>% select(from, to),  
 directed = FALSE,  
 vertices = nodes\_tbl %>% transmute(name = as.character(idx))  
)  
  
# ─── 3. Find the chosen person’s idx (character) ────────────────────────────  
center\_idx\_char <- nodes\_tbl %>%  
 filter(name == center\_person, `Node Type` == "Person") %>%  
 pull(idx) %>%  
 as.character()  
if (length(center\_idx\_char) != 1) {  
 stop("Person '", center\_person, "' not found or not unique.")  
}  
  
# ─── 4. Compute the ego‐set of vertices within max\_hops of the center ────────  
ego\_list <- ego(g\_all, order = max\_hops, nodes = center\_idx\_char, mode = "all")[[1]]  
ego\_indices <- as.integer(V(g\_all)$name[ego\_list]) # numeric idx of all nodes in ego‐set  
  
# ─── 5. Filter edges so that both endpoints lie in that ego‐set AND edge‐type is in keep\_edge\_types ─  
ego\_edges <- edges\_idx %>%  
 filter(  
 from %in% ego\_indices,  
 to %in% ego\_indices,  
 `Edge Type` %in% keep\_edge\_types  
 )  
  
if (nrow(ego\_edges) == 0) {  
 message("No edges remain when limiting to ", max\_hops, " hop(s) around ", center\_person, ".")  
} else {  
 # ─── 6. Build the list of involved nodes in this ego‐set ─────────────────────  
 ego\_node\_ids <- unique(c(ego\_edges$from, ego\_edges$to))  
 ego\_nodes <- nodes\_tbl %>% slice(ego\_node\_ids)  
   
 # ─── 7. Construct an igraph for this ego network ───────────────────────────  
 vertices\_df <- ego\_nodes %>%  
 transmute(name = as.character(idx), label = name, type = `Node Type`)  
 edges\_df <- ego\_edges %>%  
 transmute(from = as.character(from), to = as.character(to), etype = `Edge Type`)  
   
 g\_ego <- graph\_from\_data\_frame(  
 d = edges\_df %>% select(from, to),  
 directed = FALSE,  
 vertices = vertices\_df  
 )  
   
 # ─── 8. Compute betweenness centrality for every vertex ────────────────────  
 bc\_vals <- betweenness(g\_ego, directed = FALSE, normalized = TRUE)  
 centrality\_tbl <- tibble(  
 idx = as.integer(names(bc\_vals)),  
 betweenness = unname(bc\_vals)  
 ) %>%  
 left\_join(  
 ego\_nodes %>% select(idx, NodeName = name, NodeType = `Node Type`),  
 by = "idx"  
 ) %>%  
 arrange(desc(betweenness))  
   
 # ─── 9. Bar‐chart of top 10 nodes by betweenness ──────────────────────────  
 centrality\_tbl %>%  
 slice\_head(n = 10) %>%  
 ggplot(aes(x = fct\_reorder(NodeName, betweenness), y = betweenness, fill = NodeType)) +  
 geom\_col(show.legend = TRUE) +  
 coord\_flip() +  
 labs(  
 title = paste0("Top 10 Nodes by Betweenness (", max\_hops, "-hop around ", center\_person, ")"),  
 x = NULL,  
 y = "Betweenness (normalized)"  
 ) +  
 theme\_minimal(base\_size = 9) +  
 theme(  
 plot.title = element\_text(size = 10, face = "bold"),  
 axis.title.y = element\_text(size = 5),  
 axis.title.x = element\_text(size = 10)  
 )  
}



|  |
| --- |
| People Influence |
| These plots depict that each artist has a difference in which external node (song/album/person) had the most influence and control. |

## Additional: Affliation works with Musical Groups

* I want to visualise the categories for Kimberly Snyder’s Sonic Renegades and their members.
* Only Kimberly Snyder has an affliated musicalgroup as Leyla Graf-Gotthard and Szymon Pyć are supposedly solo artists as there isnt any nodes linking them to a group.

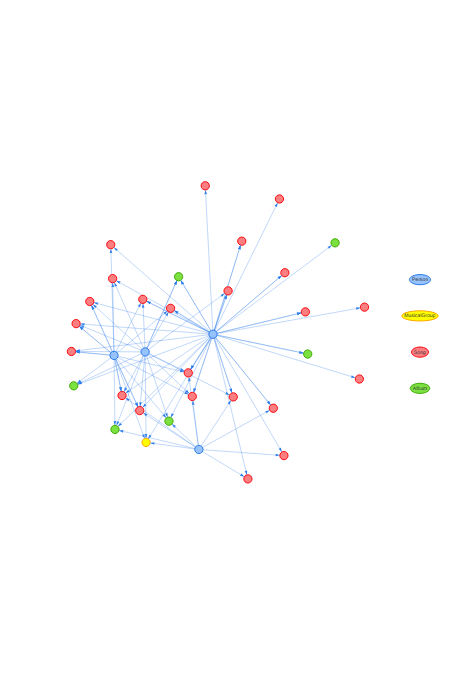
### Kimberly Snyder and Sonic Renegade members

Following factors are used to visualize Kimberly and her members.

* “MemberOf”, “ComposerOf”, “PerformerOf”,“LyricistOf”, “RecordedBy”,“ProducerOf”,“DistributedBy”

#### MusicalGroup Linkages

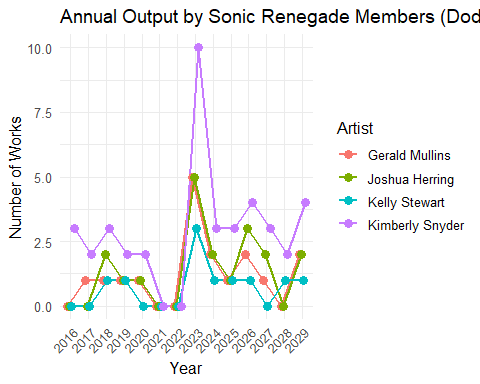
#── Ingest your graph data ────────────────────────────────────────────────────  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes)  
edges\_tbl <- as\_tibble(kg\_raw$links)  
  
#── Map source/target to row indices ──────────────────────────────────────────  
id\_map <- nodes\_tbl %>%   
 mutate(index = row\_number()) %>%   
 select(id, index)  
  
edges\_idx <- edges\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = index) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = index) %>%  
 filter(!is.na(from) & !is.na(to))  
  
#── Define your five Sonic Renegade members ────────────────────────────────────────  
artists <- c(  
 "Kimberly Snyder",  
 "Gerald Mullins",  
 "Kelly Stewart",  
 "Joshua Herring"  
)  
  
#── Pull only the relevant edges for those artists ────────────────────────────  
career\_edges <- edges\_idx %>%  
 filter(`Edge Type` %in% c("MemberOf", "ComposerOf", "PerformerOf","LyricistOf", "RecordedBy","ProducerOf","DistributedBy")) %>%  
 filter(  
 from %in% which(nodes\_tbl$name %in% artists) |  
 to %in% which(nodes\_tbl$name %in% artists)  
 )  
  
#── Slice out all involved nodes ──────────────────────────────────────────────  
career\_node\_ids <- unique(c(career\_edges$from, career\_edges$to))  
career\_nodes <- nodes\_tbl %>% slice(career\_node\_ids)  
  
#── Build visNetwork nodes df (uniform sizing) ───────────────────────────────  
vn\_nodes <- career\_nodes %>%  
 mutate(  
 id = row\_number(),  
 label = name,  
 group = `Node Type`,  
 value = 1  
 )  
  
#── Remap edges to the new vn\_node ids ────────────────────────────────────────  
id\_lookup <- tibble(old = career\_node\_ids, new = vn\_nodes$id)  
  
vn\_edges <- career\_edges %>%  
 inner\_join(id\_lookup, by = c("from" = "old")) %>%  
 select(-from) %>% rename(from = new) %>%  
 inner\_join(id\_lookup, by = c("to" = "old")) %>%  
 select(-to) %>% rename(to = new) %>%  
 select(from, to, title = `Edge Type`)  
  
#── Plot ─────────────────────────────────────────────────────────────────────  
visNetwork(vn\_nodes, vn\_edges, height = "600px", width = "100%") %>%  
 visNodes(shape = "dot", scaling = list(min = 5, max = 30)) %>%  
 visEdges(arrows = "to") %>%  
 visLegend(width = 0.1, position = "right") %>%  
 visIgraphLayout(layout = "layout\_with\_fr", randomSeed = 1234) %>%  
 visPhysics(enabled = FALSE)



#### Timeline of Sonic Renegade’ members works

* I would like to view the timeline of the works by each of the members to see who has been having consistent work throughout the years.

library(lubridate)  
library(tidyr) # for unnest\_wider()  
  
# 1. Load & index graph (as before) -----------------------------  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes)  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
ivy\_id\_map <- nodes\_tbl %>%   
 mutate(ivy\_row = row\_number()) %>%   
 select(id, ivy\_row)  
  
ivy\_edges <- links\_tbl %>%  
 left\_join(ivy\_id\_map, by = c("source" = "id")) %>% rename(from = ivy\_row) %>%  
 left\_join(ivy\_id\_map, by = c("target" = "id")) %>% rename(to = ivy\_row) %>%  
 filter(!is.na(from), !is.na(to))  
  
# 2. Get Ivy Echos members ---------------------------------------  
band\_idx <- which(nodes\_tbl$name == "Sonic Renegade")  
member\_idxs <- ivy\_edges %>%  
 filter(`Edge Type` == "MemberOf", to == band\_idx) %>%  
 pull(from)  
member\_names <- nodes\_tbl$name[member\_idxs]  
  
# 3. Filter to their ComposerOf/PerformerOf and extract years ----  
release\_edges <- ivy\_edges %>%  
 filter(`Edge Type` %in% c("ComposerOf","PerformerOf","LyricistOf", "RecordedBy","ProducerOf","DistributedBy"),  
 from %in% member\_idxs)  
  
release\_data <- release\_edges %>%  
 mutate(  
 Artist = nodes\_tbl$name[from],  
 Year = as.integer(nodes\_tbl$release\_date[to])  
 ) %>%  
 filter(!is.na(Year))  
  
# 4. Summarize & “complete” zero years ---------------------------  
raw\_counts <- release\_data %>%  
 count(Artist, Year)  
  
# determine full span of years in your dataset  
year\_span <- seq(min(raw\_counts$Year), max(raw\_counts$Year))  
  
timeline\_tbl <- raw\_counts %>%  
 complete(  
 Artist = member\_names,  
 Year = year\_span,  
 fill = list(n = 0)  
 )  
  
# 5. Plot with zeros shown ----------------------------------------  
ggplot(timeline\_tbl %>% mutate(Year\_f = factor(Year)),   
 aes(x = Year\_f, y = n, color = Artist, group = Artist)) +  
 geom\_line(position = position\_dodge(width = 0.5), size = 1) +  
 geom\_point(position = position\_dodge(width = 0.5), size = 3) +  
 scale\_x\_discrete(drop = FALSE) +  
 labs(  
 title = "Annual Output by Sonic Renegade Members (Dodged)",  
 x = "Year",  
 y = "Number of Works",  
 color = "Artist"  
 ) +  
 theme\_minimal(base\_size = 12) +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))



|  |
| --- |
| Additional |
| Based on the above plot, the trend of how many songs released by each member is similar. However, Kimberly Snyder has more songs each year as compared to the other group members.  This could also show how Kimberly had more notable songs due to more effort in producing songs each year which lead to her immense popularity and influence. |

## **Summary of part (a)**

1. Notable songs and release across the years are important factors for artist’s popularity.
2. External factors such as influences factors show how much control and spread of the artist’s information is spread across to people.

# Question 3 Part (b)

Part (b): **Using this characterization, give three predictions of who the next Oceanus Folk stars with be over the next five years**

To predict the next Oceanus Folk stars over the next 5 years, I would need to check on the popularity + influence nodes that was done in part (a). I will see if there is a pattern in the plots and predict from there who I think will be the next upcoming stars.

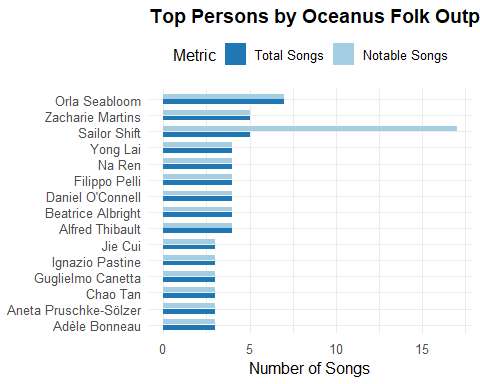
|  |
| --- |
| Factors to consider |
| 1. Count of Notable songs 2. Years since 1st release song (Limiting the years since the artist’s first released song”) 3. Betweenness Centrality score in Oceanus Folk genre 4. Similar pattern to Top Stars in industry 5. Advisable to check artists as individuals and not between group members as some might be an older veteran in the industry |

## Popularity

### Number of Notable songs of Oceanus Folk

* Let’s take a look at all the songs + notable songs released by Oceanus Folk throughout their careers.
* I will do a global count of every Oceanus Folk song marked “notable == True” to have an overview without any restrictions.
* I will plot the total songs released beside the notable songs barchart as comparison.

# 1. Load & index the graph  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>%   
 mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
id\_map <- nodes\_tbl %>% select(id, idx)  
  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from) & !is.na(to))  
  
# 2. Identify all Person nodes  
person\_idxs <- which(nodes\_tbl$`Node Type` == "Person")  
  
# 3a. Count all Oceanus Folk songs per Person  
total\_oceanus\_tbl <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf", "PerformerOf"),  
 from %in% person\_idxs) %>%  
 left\_join(  
 nodes\_tbl %>% select(idx, genre, `Node Type`),  
 by = c("to" = "idx")  
 ) %>%  
 filter(`Node Type` == "Song", genre == "Oceanus Folk") %>%  
 mutate(Person = nodes\_tbl$name[from]) %>%  
 count(Person, name = "TotalOceanusSongs")  
  
# 3b. Count only notable Oceanus Folk songs per Person  
notable\_oceanus\_tbl <- edges\_idx %>%  
 filter(`Edge Type` %in% c("ComposerOf", "PerformerOf"),  
 from %in% person\_idxs) %>%  
 left\_join(  
 nodes\_tbl %>% select(idx, notable, genre, `Node Type`),  
 by = c("to" = "idx")  
 ) %>%  
 filter(`Node Type` == "Song",  
 genre == "Oceanus Folk",  
 notable == TRUE) %>%  
 mutate(Person = nodes\_tbl$name[from]) %>%  
 count(Person, name = "NotableOceanusSongs")  
  
# 4. Join total & notable counts, replace missing with 0  
combined\_tbl <- full\_join(total\_oceanus\_tbl, notable\_oceanus\_tbl, by = "Person") %>%  
 replace\_na(list(TotalOceanusSongs = 0, NotableOceanusSongs = 0)) %>%  
 arrange(desc(NotableOceanusSongs))  
  
# 5. Take top 15 by Notable count, then reorder factor by Notable  
top\_n <- 15  
plot\_tbl <- combined\_tbl %>%  
 slice\_head(n = top\_n) %>%  
 mutate(Person = fct\_reorder(Person, NotableOceanusSongs))  
  
# 6. Pivot to long form for grouped/stacked bar plotting  
plot\_long <- plot\_tbl %>%  
 pivot\_longer(cols = c(TotalOceanusSongs, NotableOceanusSongs),  
 names\_to = "Metric", values\_to = "Count")  
  
# 7. Plot side-by-side columns: Total vs Notable  
ggplot(plot\_long, aes(x = Person, y = Count, fill = Metric)) +  
 geom\_col(position = position\_dodge(width = 0.7), width = 0.6) +  
 coord\_flip() +  
 scale\_fill\_manual(  
 values = c(  
 TotalOceanusSongs = "#a6cee3",  
 NotableOceanusSongs = "#1f78b4"  
 ),  
 labels = c("Total Songs", "Notable Songs")  
 ) +  
 labs(  
 title = "Top Persons by Oceanus Folk Output (Total vs. Notable)",  
 x = NULL,  
 y = "Number of Songs",  
 fill = "Metric"  
 ) +  
 theme\_minimal(base\_size = 12) +  
 theme(  
 legend.position = "top",  
 axis.text.y = element\_text(size = 10),  
 plot.title = element\_text(size = 14, face = "bold")  
 )

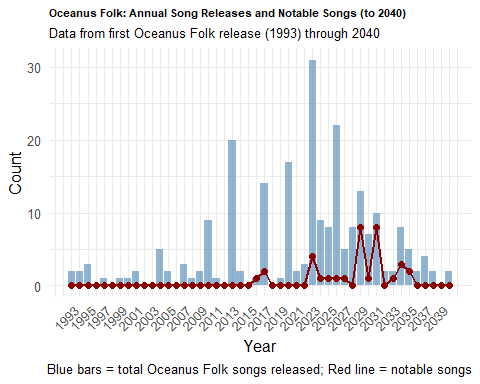


|  |
| --- |
| Overall Artists in Oceanus Folk |
| You can see the spread of artists with notable songs but we have yet to limit the years as we are not interested in long-careered artists but **rising artists**. |

### Songs timeline

I will plot these songs across the years to visualize which years we should set as our limit.

library(lubridate)  
  
# 1. Load & index the graph  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>%  
 mutate(idx = row\_number())  
  
# 2. Filter to Oceanus Folk Song nodes and extract their year + notable flag  
oceanus\_songs <- nodes\_tbl %>%  
 filter(`Node Type` == "Song", genre == "Oceanus Folk") %>%  
 mutate(  
 # Parse release\_date and written\_date to integers (YYYY)  
 release\_year = suppressWarnings(as.integer(str\_extract(release\_date, "\\d{4}"))),  
 written\_year = suppressWarnings(as.integer(str\_extract(written\_date, "\\d{4}"))),  
 # Choose the earliest non-NA year as the official song\_year  
 song\_year = pmin(release\_year, written\_year, na.rm = TRUE),  
 # Flag whether this song is “notable” (has a notoriety\_date)  
 is\_notable = !is.na(notoriety\_date)  
 ) %>%  
 # Only keep those with a valid song\_year ≤ 2040  
 filter(!is.na(song\_year), song\_year <= 2040) %>%  
 select(idx, song\_year, is\_notable)  
  
# 3. Determine the earliest Oceanus Folk “debut” year  
debut\_year <- min(oceanus\_songs$song\_year, na.rm = TRUE)  
  
# 4. Build a data frame of year-by-year counts from debut\_year through 2040  
year\_seq <- tibble(year = seq(debut\_year, 2040))  
  
yearly\_counts <- oceanus\_songs %>%  
 group\_by(song\_year) %>%  
 summarize(  
 total\_releases = n(),  
 total\_notable = sum(is\_notable),  
 .groups = "drop"  
 ) %>%  
 rename(year = song\_year)  
  
# 5. Left-join to ensure we include years with zero releases/notable  
yearly\_trends <- year\_seq %>%  
 left\_join(yearly\_counts, by = "year") %>%  
 replace\_na(list(total\_releases = 0, total\_notable = 0))  
  
# 6. Plot both series on the same plot  
ggplot(yearly\_trends, aes(x = year)) +  
 geom\_col(aes(y = total\_releases), fill = "steelblue", alpha = 0.6) +  
 geom\_line(aes(y = total\_notable), color = "darkred", size = 1) +  
 geom\_point(aes(y = total\_notable), color = "darkred", size = 2) +  
 scale\_x\_continuous(breaks = seq(debut\_year, 2040, by = 2)) +  
 labs(  
 title = "Oceanus Folk: Annual Song Releases and Notable Songs (to 2040)",  
 subtitle = paste0("Data from first Oceanus Folk release (", debut\_year, ") through 2040"),  
 x = "Year",  
 y = "Count",  
 caption = "Blue bars = total Oceanus Folk songs released; Red line = notable songs"  
 ) +  
 theme\_minimal(base\_size = 12) +  
 theme(  
 plot.title = element\_text(size = 8, face = "bold"),  
 plot.subtitle = element\_text(size = 10),  
 axis.text.x = element\_text(angle = 45, hjust = 1)  
 )

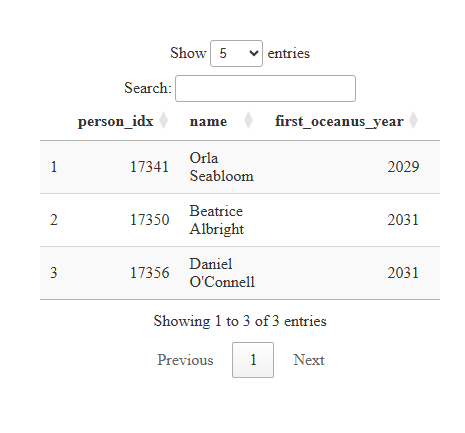


|  |
| --- |
| Timeline Note |
| * There weren’t any notable songs released by the Oceanus Folk artists from 2036 to 2040. * Based on this, I will want to limit the spread to around 10 years since the last notable song in 2035. * The range I will select to determine my 3 rising stars will be from **2025 to 2035**. |

## Direct Prediction based on notable songs

* Below is a direct prediction on the **top 3 “next Oceanus Folk stars”** based on limiting the notable songs and release years from 2025 to 2035.

library(DT)  
library(lubridate)  
  
# ─── 0. PARAMETERS ─────────────────────────────────────────────────────────────  
window\_start <- 2025  
window\_end <- 2035  
early\_window\_years <- 5  
influence\_edge\_types <- c(  
 "InterpolatesFrom",  
 "CoverOf",  
 "LyricalReferenceTo",  
 "DirectlySamples",  
 "InStyleOf"  
)  
  
# ─── 1. LOAD & PREPARE NODES + EDGES ───────────────────────────────────────────  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>%  
 mutate(idx = row\_number()) # numeric index for joining  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from), !is.na(to))  
  
# ─── 2. EXTRACT “OCEANUS FOLK” SONGS & THEIR FIRST‐RELEASE YEARS ─────────────  
songs\_oceanus <- nodes\_tbl %>%  
 filter(`Node Type` == "Song", genre == "Oceanus Folk") %>%  
 mutate(  
 release\_year = suppressWarnings(as.integer(str\_extract(release\_date, "\\d{4}"))),  
 written\_year = suppressWarnings(as.integer(str\_extract(written\_date, "\\d{4}"))),  
 first\_year = pmin(release\_year, written\_year, na.rm = TRUE)  
 ) %>%  
 filter(!is.na(first\_year)) %>%  
 filter(first\_year >= window\_start, first\_year <= window\_end) %>%  
 select(song\_idx = idx, first\_year)  
  
person\_idxs <- nodes\_tbl %>%  
 filter(`Node Type` == "Person") %>%  
 pull(idx)  
  
person\_song\_edges <- edges\_idx %>%  
 filter(  
 `Edge Type` %in% c("ComposerOf", "PerformerOf"),  
 from %in% person\_idxs,  
 to %in% songs\_oceanus$song\_idx  
 ) %>%  
 select(person\_idx = from, song\_idx = to)  
  
person\_first\_oceanus <- person\_song\_edges %>%  
 left\_join(songs\_oceanus, by = "song\_idx") %>%  
 group\_by(person\_idx) %>%  
 summarise(first\_oceanus\_year = min(first\_year, na.rm = TRUE), .groups = "drop") %>%  
 filter(first\_oceanus\_year >= window\_start, first\_oceanus\_year <= window\_end)  
  
if (nrow(person\_first\_oceanus) == 0) {  
 stop("No emerging Oceanus Folk artists found between ", window\_start, " and ", window\_end, ".")  
}  
  
# ─── 3. COMPUTE EARLY‐CAREER “POPULARITY” ─────────────────────────────────────  
songs\_oceanus\_full <- nodes\_tbl %>%  
 filter(`Node Type` == "Song", genre == "Oceanus Folk") %>%  
 mutate(  
 release\_year = suppressWarnings(as.integer(str\_extract(release\_date, "\\d{4}"))),  
 written\_year = suppressWarnings(as.integer(str\_extract(written\_date, "\\d{4}"))),  
 first\_year\_song = pmin(release\_year, written\_year, na.rm = TRUE),  
 is\_notable = !is.na(notoriety\_date)  
 ) %>%  
 select(song\_idx = idx, first\_year\_song, is\_notable)  
  
artist\_songs <- edges\_idx %>%  
 filter(  
 `Edge Type` %in% c("ComposerOf", "PerformerOf"),  
 from %in% person\_first\_oceanus$person\_idx,  
 to %in% songs\_oceanus\_full$song\_idx  
 ) %>%  
 select(person\_idx = from, song\_idx = to) %>%  
 left\_join(songs\_oceanus\_full, by = "song\_idx") %>%  
 left\_join(person\_first\_oceanus, by = "person\_idx")  
  
artist\_songs\_early <- artist\_songs %>%  
 filter(  
 first\_year\_song >= first\_oceanus\_year,  
 first\_year\_song <= first\_oceanus\_year + early\_window\_years  
 )  
  
popularity\_tbl <- artist\_songs\_early %>%  
 filter(is\_notable) %>%  
 count(person\_idx, name = "early\_notable\_songs")  
  
popularity\_tbl <- person\_first\_oceanus %>%  
 select(person\_idx) %>%  
 left\_join(popularity\_tbl, by = "person\_idx") %>%  
 mutate(early\_notable\_songs = replace\_na(early\_notable\_songs, 0))  
  
# ─── 4. COMPUTE EARLY‐CAREER “INFLUENCE” ─────────────────────────────────────  
nodes\_years <- nodes\_tbl %>%  
 mutate(  
 release\_year = suppressWarnings(as.integer(str\_extract(release\_date, "\\d{4}"))),  
 written\_year = suppressWarnings(as.integer(str\_extract(written\_date, "\\d{4}"))),  
 notoriety\_year = suppressWarnings(as.integer(str\_extract(notoriety\_date, "\\d{4}"))),  
 node\_year = pmin(release\_year, written\_year, notoriety\_year, na.rm = TRUE)  
 ) %>%  
 select(idx, node\_year)  
  
artist\_infl\_edges <- edges\_idx %>%  
 filter(  
 `Edge Type` %in% influence\_edge\_types,  
 (from %in% person\_first\_oceanus$person\_idx) |  
 (to %in% person\_first\_oceanus$person\_idx)  
 ) %>%  
 left\_join(nodes\_years, by = c("from" = "idx")) %>% rename(year\_from = node\_year) %>%  
 left\_join(nodes\_years, by = c("to" = "idx")) %>% rename(year\_to = node\_year) %>%  
 mutate(edge\_year = pmax(year\_from, year\_to, na.rm = TRUE))  
  
artist\_infl\_early <- artist\_infl\_edges %>%  
 mutate(person\_idx = if\_else(from %in% person\_first\_oceanus$person\_idx, from, to)) %>%  
 left\_join(person\_first\_oceanus, by = "person\_idx") %>%  
 filter(  
 edge\_year >= first\_oceanus\_year,  
 edge\_year <= first\_oceanus\_year + early\_window\_years  
 ) %>%  
 count(person\_idx, name = "early\_influence\_edges")  
  
influence\_tbl <- person\_first\_oceanus %>%  
 select(person\_idx) %>%  
 left\_join(artist\_infl\_early, by = "person\_idx") %>%  
 mutate(early\_influence\_edges = replace\_na(early\_influence\_edges, 0))  
  
# ─── 5. COMBINE & RANK EMERGING ARTISTS ───────────────────────────────────────  
emerging\_metrics <- person\_first\_oceanus %>%  
 left\_join(popularity\_tbl, by = "person\_idx") %>%  
 left\_join(influence\_tbl, by = "person\_idx") %>%  
 left\_join(nodes\_tbl %>% select(person\_idx = idx, name), by = "person\_idx") %>%  
 mutate(total\_score = early\_notable\_songs + early\_influence\_edges) %>%  
 select(person\_idx, name, first\_oceanus\_year,  
 early\_notable\_songs, early\_influence\_edges, total\_score) %>%  
 arrange(desc(total\_score), desc(early\_notable\_songs))  
  
top3\_emerging <- emerging\_metrics %>% slice\_head(n = 3)  
  
# ─── 6. RENDER AS DATA TABLE WITH SCROLLING ──────────────────────────────────  
DT::datatable(  
 top3\_emerging,  
 extensions = "Buttons",  
 options = list(  
 scrollX = TRUE,  
 pageLength = 5  
 )  
)

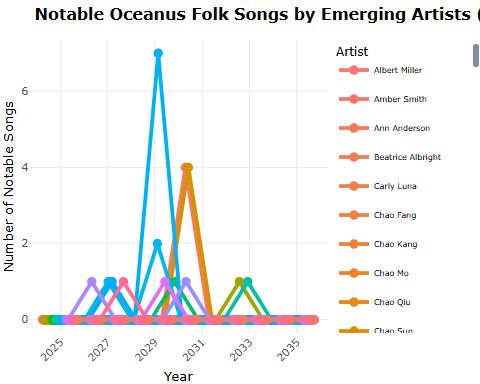


|  |
| --- |
| Most notable songs from emerging artists in Oceanus Folk genre |
| * Orla Seabloom, Beatrice Albright & Daniel O’Connell have the most notable songs (4<=) and their debut was between 2025 to 2035. * Let’s visualise them against the rest of the artist to have a clearer view on this take. |

## Visualising Notable Oceanus Folk songs based on debut of Artist from 2025 to 2035

* I will be restricting the artists whose first Oceanus Folk release fell in 2025–2035 and count that as notable songs for more relevance for recent years.

library(lubridate)  
library(plotly)  
  
# 1. Load & index nodes and edges  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>%   
 mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
# Build numeric from/to indices for each edge  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from), !is.na(to))  
  
# 2. Extract Oceanus Folk songs and compute 'song\_year' + notable flag  
oceanus\_songs\_full <- nodes\_tbl %>%  
 filter(`Node Type` == "Song", genre == "Oceanus Folk") %>%  
 mutate(  
 release\_year = suppressWarnings(as.integer(str\_extract(release\_date, "\\d{4}"))),  
 written\_year = suppressWarnings(as.integer(str\_extract(written\_date, "\\d{4}"))),  
 song\_year = pmin(release\_year, written\_year, na.rm = TRUE),  
 is\_notable = !is.na(notoriety\_date)  
 ) %>%  
 filter(!is.na(song\_year)) %>%  
 select(song\_idx = idx, song\_year, is\_notable)  
  
# 3. Identify all Person indices  
person\_idxs <- nodes\_tbl %>%  
 filter(`Node Type` == "Person") %>%  
 pull(idx)  
  
# 4. Find ComposerOf / PerformerOf edges linking persons → Oceanus Folk songs  
person\_song\_edges <- edges\_idx %>%  
 filter(  
 `Edge Type` %in% c("ComposerOf", "PerformerOf"),  
 from %in% person\_idxs,  
 to %in% oceanus\_songs\_full$song\_idx  
 ) %>%  
 select(person\_idx = from, song\_idx = to)  
  
# 5. Compute each person’s first Oceanus Folk year, then filter to 2028–2040  
person\_first\_oceanus <- person\_song\_edges %>%  
 left\_join(oceanus\_songs\_full, by = "song\_idx") %>%  
 group\_by(person\_idx) %>%  
 summarize(  
 first\_oceanus\_year = min(song\_year, na.rm = TRUE),  
 .groups = "drop"  
 ) %>%  
 filter(first\_oceanus\_year >= 2025, first\_oceanus\_year <= 2035) %>%  
 # Attach PersonName  
 left\_join(  
 nodes\_tbl %>% select(person\_idx = idx, PersonName = name),  
 by = "person\_idx"  
 )  
  
if (nrow(person\_first\_oceanus) == 0) {  
 stop("No Oceanus Folk persons with first release between 2028 and 2040.")  
}  
  
# 6. For those emerging persons, find their notable Oceanus Folk songs between 2028–2040  
artist\_notable\_songs <- edges\_idx %>%  
 filter(  
 `Edge Type` %in% c("ComposerOf", "PerformerOf"),  
 from %in% person\_first\_oceanus$person\_idx,  
 to %in% oceanus\_songs\_full$song\_idx  
 ) %>%  
 left\_join(oceanus\_songs\_full, by = c("to" = "song\_idx")) %>%  
 left\_join(person\_first\_oceanus, by = c("from" = "person\_idx")) %>%  
 # keep only songs that are notable and within 2028–2040  
 filter(  
 is\_notable,  
 song\_year >= 2025,  
 song\_year <= 2035  
 ) %>%  
 select(PersonName, song\_year)  
  
# 7. Count notable songs per PersonName per year  
notable\_counts <- artist\_notable\_songs %>%  
 count(PersonName, year = song\_year, name = "NotableCount")  
  
# 8. Create a year grid from 2028 to 2040 and person list  
year\_seq <- tibble(year = 2025:2035)  
person\_list <- person\_first\_oceanus %>% pull(PersonName) %>% unique()  
  
# Expand to all combinations and fill missing with zero  
all\_combos <- expand\_grid(PersonName = person\_list, year = 2025:2035)  
  
notable\_trends <- all\_combos %>%  
 left\_join(notable\_counts, by = c("PersonName", "year")) %>%  
 replace\_na(list(NotableCount = 0))  
  
# 9. Static ggplot (for sizing/legend adjustments)  
static\_plot <- ggplot(notable\_trends, aes(x = year, y = NotableCount, color = PersonName, group = PersonName)) +  
 geom\_line(position = position\_dodge(width = 1.5), size = 1) +  
 geom\_point(position = position\_dodge(width = 1.5), size = 2) +  
 scale\_x\_continuous(breaks = seq(2025, 2035, by = 2)) +  
 labs(  
 title = "Notable Oceanus Folk Songs by Emerging Artists (2025–2035)",  
 subtitle = "Click a legend item to isolate that artist",  
 x = "Year",  
 y = "Number of Notable Songs",  
 color = "Artist"  
 ) +  
 theme\_minimal(base\_size = 10) +  
 theme(  
 legend.text = element\_text(size = 8),  
 legend.title = element\_text(size = 9),  
 plot.title = element\_text(size = 12, face = "bold"),  
 plot.subtitle = element\_text(size = 10),  
 axis.text.x = element\_text(angle = 45, hjust = 1)  
 )  
  
# 10. Convert to interactive Plotly, adjust legend for “toggle others” on click  
interactive\_plot <- ggplotly(static\_plot, tooltip = c("x", "y", "colour")) %>%  
 layout(  
 legend = list(  
 font = list(size = 8),  
 itemclick = "toggleothers", # clicking a legend item isolates that trace  
 itemdoubleclick = "toggle" # double-click brings back all traces  
 )  
 )  
  
# 11. Print the interactive plot  
interactive\_plot



|  |
| --- |
| Findings based on Notable Songs |
| * This plot shows that Orla, Beatrice and Daniel indeed have the most notable songs in the “recent” time period. |

|  |
| --- |
| Sailor Shift outlier |
| * It’s worth to note that Sailor Shift has alot more notable songs overall. But only 1 of it falls under the 2025 - 2035 period. * The reason why Sailor Shift isn’t considered an **“rising star”** is due to the combination of 2 filters.   1. Emerging-artist filter (2025 - 2035) - **1st** Oceanus Folk song\_year must fall in this window.   2. “Notable” + Year - The song must have a non-NA notoriety\_date in the period. |

**Orla Seabloom, Beatrice Albright and Daniel O’Connell** are the most promising upcoming Oceanus Folk artists based on popularity in recent years.

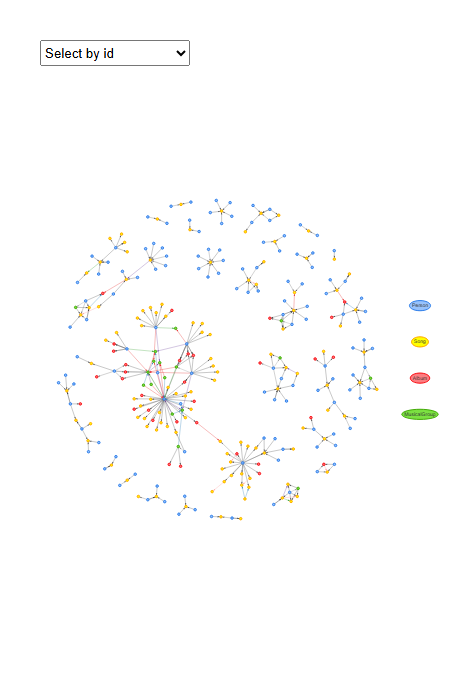
## Influences

* I will plot the influence diagram without splitting the work and influence related edges. This will show a good overview of Orla Seabloom, Beatrice Albright and Daniel O’Connell’s connection.

### Oceanus Folk visNetwork from 2025 - 2035

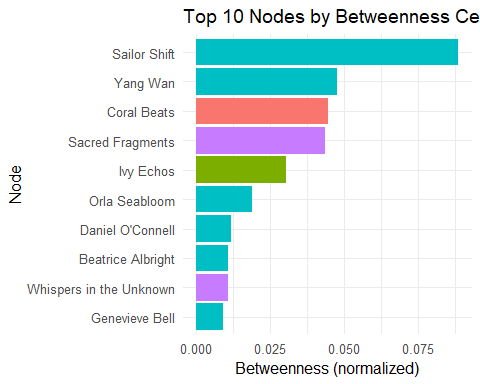
* Limiting the hop to 1.

# ─── 0. How many hops? ────────────────────────────────────────────────────────  
max\_hops <- 1  
  
# ─── 1. Load & index all nodes and edges ────────────────────────────────────  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>% mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from) & !is.na(to))  
  
# ─── 2. Extract Oceanus Folk songs with their earliest year ────────────────  
oceanus\_songs\_full <- nodes\_tbl %>%  
 filter(`Node Type` == "Song", genre == "Oceanus Folk") %>%  
 mutate(  
 release\_year = suppressWarnings(as.integer(str\_extract(release\_date, "\\d{4}"))),  
 written\_year = suppressWarnings(as.integer(str\_extract(written\_date, "\\d{4}"))),  
 song\_year = pmin(release\_year, written\_year, na.rm = TRUE)  
 ) %>%  
 filter(!is.na(song\_year)) %>%  
 select(song\_idx = idx, song\_year)  
  
# ─── 3. Identify Person→Oceanus Folk edges ─────────────────────────────────  
person\_idxs <- nodes\_tbl %>%  
 filter(`Node Type` == "Person") %>%  
 pull(idx)  
  
person\_song\_edges <- edges\_idx %>%  
 filter(  
 `Edge Type` %in% c("ComposerOf", "PerformerOf"),  
 from %in% person\_idxs,  
 to %in% oceanus\_songs\_full$song\_idx  
 )  
  
# ─── 4. Compute each person’s first Oceanus Folk year, then filter to 2025–2035 ─  
person\_first\_oceanus <- person\_song\_edges %>%  
 left\_join(oceanus\_songs\_full, by = c("to" = "song\_idx")) %>%  
 group\_by(from) %>%  
 summarize(first\_oceanus\_year = min(song\_year, na.rm = TRUE), .groups = "drop") %>%  
 filter(first\_oceanus\_year >= 2025, first\_oceanus\_year <= 2035) %>%  
 rename(person\_idx = from)  
  
if (nrow(person\_first\_oceanus) == 0) {  
 stop("No Oceanus Folk artists with first release between 2025 and 2035.")  
}  
  
focus\_artists <- nodes\_tbl %>%  
 filter(idx %in% person\_first\_oceanus$person\_idx, `Node Type` == "Person") %>%  
 pull(name)  
  
# ─── 5. Build an undirected igraph of the full graph ────────────────────────  
g\_all <- graph\_from\_data\_frame(  
 d = edges\_idx %>% select(from, to),  
 directed = FALSE,  
 vertices = nodes\_tbl %>% transmute(name = as.character(idx))  
)  
  
# ─── 6. Compute each focus artist’s 1-hop ego set; then union ─────────────  
ego\_indices\_list <- lapply(person\_first\_oceanus$person\_idx, function(a\_idx) {  
 ego(g\_all, order = max\_hops, nodes = as.character(a\_idx), mode = "all")[[1]]  
})  
all\_ego\_numeric <- unique(as.integer(unlist(ego\_indices\_list)))  
  
# ─── 7. Filter edges to include both influence and work edges among that union ─  
keep\_edge\_types <- c(  
 "InterpolatesFrom", "InStyleOf", "LyricalReferenceTo",  
 "CoverOf", "DirectlySamples",  
 "ComposerOf", "PerformerOf", "MemberOf", "Released"  
)  
  
career\_edges <- edges\_idx %>%  
 filter(  
 from %in% all\_ego\_numeric,  
 to %in% all\_ego\_numeric,  
 `Edge Type` %in% keep\_edge\_types  
 )  
  
if (nrow(career\_edges) == 0) {  
 stop("No matching edges found within the 1-hop neighborhoods of selected Oceanus Folk artists.")  
}  
  
# ─── 8. Build the list of all involved nodes ────────────────────────────────  
career\_node\_ids <- unique(c(career\_edges$from, career\_edges$to))  
career\_nodes <- nodes\_tbl %>% slice(career\_node\_ids) %>%  
 mutate(  
 vis\_id = row\_number(),  
 label = name,  
 group = `Node Type`,  
 title = paste0("<b>", name, "</b><br>Type: ", `Node Type`)  
 )  
  
# ─── 9. Remap each career\_edge → (from\_vis, to\_vis) by matching against career\_node\_ids ─  
career\_edges <- career\_edges %>%  
 mutate(  
 from\_vis = match(from, career\_node\_ids),  
 to\_vis = match(to, career\_node\_ids)  
 )  
  
# ─── 10. Construct visNetwork data frames for nodes & edges ─────────────────  
vn\_nodes <- career\_nodes %>%  
 transmute(  
 id = vis\_id,  
 label = label,  
 group = group,  
 title = title  
 )  
  
vn\_edges <- career\_edges %>%  
 transmute(  
 from = from\_vis,  
 to = to\_vis,  
 label = `Edge Type`,  
 title = `Edge Type`,  
 color = case\_when(  
 label == "InterpolatesFrom" ~ "#1f78b4",  
 label == "InStyleOf" ~ "#e31a1c",  
 label == "LyricalReferenceTo" ~ "#6a3d9a",  
 label == "CoverOf" ~ "#fb9a99",  
 label == "DirectlySamples" ~ "#33a02c",  
 label == "ComposerOf" ~ "#666666",  
 label == "PerformerOf" ~ "#444444",  
 label == "MemberOf" ~ "#888888",  
 label == "Released" ~ "#AAAAAA",  
 TRUE ~ "#999999"  
 )  
 )  
  
# ─── 11. Determine which vis\_id correspond to the focus artists (sorted alphabetically) ──  
focus\_vis\_ids <- vn\_nodes %>%  
 filter(label %in% focus\_artists) %>%  
 arrange(label) %>% # sort by artist name ascending  
 pull(id)  
  
# ─── 12. Render interactive visNetwork with alphabetically sorted dropdown ───  
visNetwork(vn\_nodes, vn\_edges, height = "600px", width = "100%") %>%  
 visNodes(  
 shape = "dot",  
 size = 14,  
 font = list(color = "black", size = 10)  
 ) %>%  
 visEdges(  
 arrows = "to",  
 labelHighlightBold = TRUE,  
 font = list(color = "white", size = 7)  
 ) %>%  
 visOptions(  
 highlightNearest = list(enabled = TRUE, degree = 1),  
 nodesIdSelection = list(  
 enabled = TRUE,  
 useLabels = TRUE,  
 values = focus\_vis\_ids  
 )  
 ) %>%  
 visLegend(useGroups = TRUE, width = 0.1, position = "right") %>%  
 visIgraphLayout(layout = "layout\_with\_fr", randomSeed = 1234) %>%  
 visPhysics(enabled = FALSE)



### Betweenness Centrality

# ─── PARAMETERS ───────────────────────────────────────────────────────────────  
max\_hops <- 1  
start\_year <- 2025  
end\_year <- 2035  
  
# ─── 1. Load & index all nodes and edges ────────────────────────────────────  
kg\_raw <- fromJSON("data/MC1\_graph.json")  
nodes\_tbl <- as\_tibble(kg\_raw$nodes) %>% mutate(idx = row\_number())  
links\_tbl <- as\_tibble(kg\_raw$links)  
  
id\_map <- nodes\_tbl %>% select(id, idx)  
edges\_idx <- links\_tbl %>%  
 left\_join(id\_map, by = c("source" = "id")) %>% rename(from = idx) %>%  
 left\_join(id\_map, by = c("target" = "id")) %>% rename(to = idx) %>%  
 filter(!is.na(from) & !is.na(to))  
  
# ─── 2. Identify all Oceanus Folk songs with computed year ─────────────────  
oceanus\_songs\_full <- nodes\_tbl %>%  
 filter(`Node Type` == "Song", genre == "Oceanus Folk") %>%  
 mutate(  
 release\_year = suppressWarnings(as.integer(str\_extract(release\_date, "\\d{4}"))),  
 written\_year = suppressWarnings(as.integer(str\_extract(written\_date, "\\d{4}"))),  
 song\_year = pmin(release\_year, written\_year, na.rm = TRUE)  
 ) %>%  
 filter(!is.na(song\_year)) %>%  
 select(song\_idx = idx, song\_year)  
  
# ─── 3. Find Person→Oceanus Folk edges and determine first-release year ────  
person\_idxs <- nodes\_tbl %>%  
 filter(`Node Type` == "Person") %>%  
 pull(idx)  
  
person\_song\_edges <- edges\_idx %>%  
 filter(  
 `Edge Type` %in% c("ComposerOf", "PerformerOf"),  
 from %in% person\_idxs,  
 to %in% oceanus\_songs\_full$song\_idx  
 )  
  
person\_first\_oceanus <- person\_song\_edges %>%  
 left\_join(oceanus\_songs\_full, by = c("to" = "song\_idx")) %>%  
 group\_by(from) %>%  
 summarize(first\_oceanus\_year = min(song\_year, na.rm = TRUE), .groups = "drop") %>%  
 filter(first\_oceanus\_year >= start\_year, first\_oceanus\_year <= end\_year) %>%  
 rename(person\_idx = from)  
  
if (nrow(person\_first\_oceanus) == 0) {  
 stop("No Oceanus Folk artists with first release between ", start\_year, " and ", end\_year, ".")  
}  
  
# ─── 4. Compute each focus artist’s 1-hop ego set; then take the union ─────  
g\_all <- graph\_from\_data\_frame(  
 d = edges\_idx %>% select(from, to),  
 directed = FALSE,  
 vertices = nodes\_tbl %>% transmute(name = as.character(idx))  
)  
  
ego\_indices\_list <- lapply(person\_first\_oceanus$person\_idx, function(a\_idx) {  
 ego(g\_all, order = max\_hops, nodes = as.character(a\_idx), mode = "all")[[1]]  
})  
all\_ego\_numeric <- unique(as.integer(unlist(ego\_indices\_list)))  
  
# ─── 5. Filter edges to include both influence and work edges among that union ─  
keep\_edge\_types <- c(  
 "InterpolatesFrom", "InStyleOf", "LyricalReferenceTo",  
 "CoverOf", "DirectlySamples",  
 "ComposerOf", "PerformerOf", "MemberOf", "Released"  
)  
  
career\_edges <- edges\_idx %>%  
 filter(  
 from %in% all\_ego\_numeric,  
 to %in% all\_ego\_numeric,  
 `Edge Type` %in% keep\_edge\_types  
 )  
  
if (nrow(career\_edges) == 0) {  
 stop("No matching edges found within the 1-hop neighborhoods of selected Oceanus Folk artists.")  
}  
  
# ─── 6. Build the list of all involved nodes ────────────────────────────────  
career\_node\_ids <- unique(c(career\_edges$from, career\_edges$to))  
career\_nodes <- nodes\_tbl %>% slice(career\_node\_ids) %>%  
 mutate(  
 vis\_id = row\_number(),  
 label = name,  
 group = `Node Type`,  
 title = paste0("<b>", name, "</b><br>Type: ", `Node Type`)  
 )  
  
# ─── 7. Build an igraph object for this subnetwork ──────────────────────────  
# We use original idx as the igraph 'name' attribute  
vertices\_df <- career\_nodes %>%  
 transmute(name = as.character(idx), label = label, group = `Node Type`, title = title)  
  
edges\_df <- career\_edges %>%  
 transmute(  
 from\_name = as.character(from),  
 to\_name = as.character(to),  
 etype = `Edge Type`  
 )  
  
g\_sub <- graph\_from\_data\_frame(  
 d = edges\_df %>% select(from = from\_name, to = to\_name),  
 directed = FALSE,  
 vertices = vertices\_df  
)  
  
# ─── 8. Compute betweenness centrality for every vertex ─────────────────────  
betw\_vals <- betweenness(g\_sub, v = V(g\_sub), directed = FALSE, normalized = TRUE)  
  
# Create a tidy tibble of (label, group, betweenness)  
betw\_tbl <- tibble(  
 node\_id = V(g\_sub)$name,  
 label = V(g\_sub)$label,  
 group = V(g\_sub)$group,  
 betweenness = betw\_vals  
) %>%  
 arrange(desc(betweenness))  
  
# ─── 9. Plot top 10 nodes by betweenness ────────────────────────────────────  
top\_n <- 10  
betw\_tbl %>%  
 slice\_head(n = top\_n) %>%  
 ggplot(aes(x = fct\_reorder(label, betweenness), y = betweenness, fill = group)) +  
 geom\_col(show.legend = FALSE) +  
 coord\_flip() +  
 labs(  
 title = "Top 10 Nodes by Betweenness Centrality",  
 x = "Node",  
 y = "Betweenness (normalized)"  
 ) +  
 theme\_minimal(base\_size = 12)



|  |
| --- |
| Influence Notes |
| Based on the plots and betweenness centrality values:   * Orla Seabloom, Beatrice Albright & Daniel O’Connell appear in the top 10 most critical nodes under Oceanus Folk in the later years (2025 - 2035) * We have to count out Sailor Shift and Yang Wan as they might have already been very popular + influential artists in the earlier years before 2025, which led them to having high scores |

## **Summary of part (b)**

Based on the Popularity and Influence plots we can predict based on Oceanus Folks who might be the upcoming rising stars in the next few years.

Looking at the plots, there is significant evidence that show that **1) Orla Seabloom, 2) Beatrice Albright & 3) Daniel O’Connell** are the next upcoming Oceanus Folk stars that debut recently that will likely continue to chart over the next 5 years.