Take-home Exercise 2

Yi Fang

2025-05-28

# 1 Overview

This take-home exercise will be done in reference to the [VAST Challenge 2025](https://vast-challenge.github.io/2025/MC1.html) and provide solutions to the first question of Mini-Challenge 1.

## 1.1 Background

One of music’s biggest superstars is Oceanus native Sailor Shift. From humble beginnings, Sailor has grown in popularity and now enjoys fans around the world. Sailor started her career on the island nation of Oceanus which can be clearly seen in her early work, she started in the genre of “Oceanus Folk”. While Sailor has moved away from the traditional Oceanus style, the Oceanus Folk has made a name for itself in the musical world. The popularity of this music is one of the factors driving an increase in tourism to a quiet island nation that used to be known for fishing.

In 2023, Sailor Shift joined the Ivy Echoes – an all-female Oceanus Folk band consisting of Sailor (vocalist), Maya Jensen (vocalist), Lila “Lilly” Hartman (guitarist), Jade Thompson (drummer), and Sophie Ramirez (bassist). They played together at venues throughout Oceanus but had broken up to pursue their individual careers by 2026. Sailor’s breakthrough came in 2028 when one of her singles went viral, launched to the top of the global charts (something no other Oceanus Folk song had ever done). Since then, she has only continued to grow in popularity worldwide.

Sailor has released a new album almost every year since her big break, and each has done better than the last. Although she has remained primarily a solo artist, she has also frequently collaborated with other established artists, especially in the Indie Pop and Indie Folk genres. She herself has branched out musically over the years but regularly returns to the Oceanus Folk genre — even as the genre’s influence on the rest of the music world has spread even more.

Sailor has always been passionate about two things: (1) spreading Oceanus Folk, and (2) helping lesser-known artists break into music. Because of those goals, she’s particularly famous for her frequent collaborations.

Additionally, because of Sailor’s success, more attention began to be paid over the years to her previous bandmates. All 4 have continued in the music industry—Maya as an independent vocalist, Lilly and Jade as instrumentalists in other bands, and Sophie as a music producer for a major record label. In various ways, all of them have contributed to the increased influence of Oceanus folk, resulting in a new generation of up-and-coming Oceanus Folk artists seeking to make a name for themselves in the music industry.

Now, as Sailor returns to Oceanus in 2040, a local journalist – Silas Reed – is writing a piece titled Oceanus Folk: Then-and-Now that aims to trace the rise of Sailor and the influence of Oceanus Folk on the rest of the music world. He has collected a large dataset of musical artists, producers, albums, songs, and influences and organized it into a knowledge graph. Your task is to help Silas create beautiful and informative visualizations of this data and uncover new and interesting information about Sailor’s past, her rise to stardom, and her influence.

## 1.2 Tasks and Questions

The objective of this take-home exercise is to address the following tasks and questions of VAST Challenge 2025’s Mini-Challenge 1.

1. Design and develop visualizations and visual analytic tools that will allow Silas to explore and understand the profile of Sailor Shift’s career
   1. Who has she been most influenced by over time?
   2. Who has she collaborated with and directly or indirectly influenced?
   3. How has she influenced collaborators of the broader Oceanus Folk community?

# 2 Getting Started

## 2.1 Load the packages

In the code chunk below, p\_load() of **pacman** package is used to load the R packages into R environemnt.

pacman::p\_load(jsonlite, tidyverse, ggtext,  
 knitr, lubridate, patchwork,  
 ggraph, tidygraph, igraph, scales,  
 ggiraph, dplyr, stringr, ggnewscale)

## 2.2 Importing Knowledge Graph Data

fromJSON() of **jsonlite** package is used to import *MC1\_graph.json* file into R and save the output object.

mc1\_data <- fromJSON("MC1/data/MC1\_graph.json")

### 2.2.1 Inspect structure

Here, str() is used to reveal the structure of *mc1\_data* object.

str(mc1\_data, 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:

## 2.3 Extracting the edges and nodes tables

Next, as\_tibble() of **tibble** package package is used to extract the nodes and links tibble data frames from *mc1\_data* object into two separate tibble data frames called *mc1\_nodes\_raw* and *mc1\_edges\_raw* respectively.

mc1\_nodes\_raw <- as\_tibble(mc1\_data$nodes)  
glimpse(mc1\_nodes\_raw)

Rows: 17,412  
Columns: 10  
$ `Node Type` <chr> "Song", "Person", "Person", "Person", "RecordLabel", "S…  
$ name <chr> "Breaking These Chains", "Carlos Duffy", "Min Qin", "Xi…  
$ single <lgl> TRUE, NA, NA, NA, NA, FALSE, NA, NA, NA, NA, TRUE, NA, …  
$ release\_date <chr> "2017", NA, NA, NA, NA, "2026", NA, NA, NA, NA, "2020",…  
$ genre <chr> "Oceanus Folk", NA, NA, NA, NA, "Lo-Fi Electronica", NA…  
$ notable <lgl> TRUE, NA, NA, NA, NA, TRUE, NA, NA, NA, NA, TRUE, NA, N…  
$ id <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1…  
$ written\_date <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, "2020", NA, NA,…  
$ stage\_name <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…  
$ notoriety\_date <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…

kable(head(mc1\_nodes\_raw, 5))

| Node Type | name | single | release\_date | genre | notable | id | written\_date | stage\_name | notoriety\_date |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Song | Breaking These Chains | TRUE | 2017 | Oceanus Folk | TRUE | 0 | NA | NA | NA |
| Person | Carlos Duffy | NA | NA | NA | NA | 1 | NA | NA | NA |
| Person | Min Qin | NA | NA | NA | NA | 2 | NA | NA | NA |
| Person | Xiuying Xie | NA | NA | NA | NA | 3 | NA | NA | NA |
| RecordLabel | Nautical Mile Records | NA | NA | NA | NA | 4 | NA | NA | NA |

mc1\_edges\_raw <- as\_tibble(mc1\_data$links)  
  
glimpse(mc1\_edges\_raw)

Rows: 37,857  
Columns: 4  
$ `Edge Type` <chr> "InterpolatesFrom", "RecordedBy", "PerformerOf", "Composer…  
$ source <int> 0, 0, 1, 1, 2, 2, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5…  
$ target <int> 1841, 4, 0, 16180, 0, 16180, 0, 5088, 14332, 11677, 2479, …  
$ key <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…

kable(head(mc1\_edges\_raw, 5))

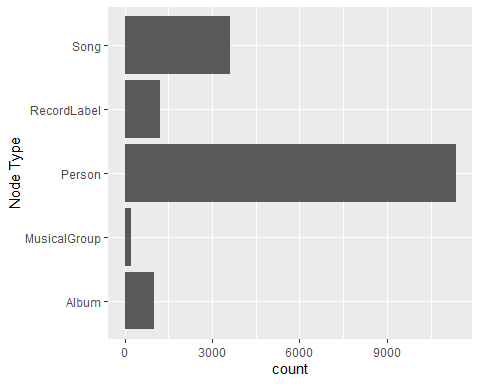
| Edge Type | source | target | key |
| --- | --- | --- | --- |
| InterpolatesFrom | 0 | 1841 | 0 |
| RecordedBy | 0 | 4 | 0 |
| PerformerOf | 1 | 0 | 0 |
| ComposerOf | 1 | 16180 | 0 |
| PerformerOf | 2 | 0 | 0 |

## 2.4 Data Overview

Before proceeding to data pre-processing, we examine the data to gain a clearer understanding of the dataset and to verify the structural integrity of the imported graph.

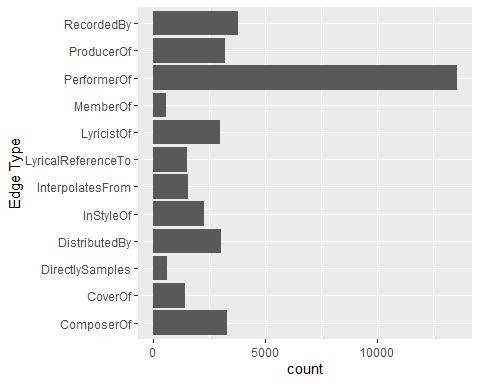
In this code chunk below, ggplot2 functions are used the reveal the frequency distribution of *Node Type* field of *mc1\_nodes\_raw*.

ggplot(data = mc1\_nodes\_raw,  
 aes(y = `Node Type`)) +  
 geom\_bar()



On the other hand, code chunk below uses ggplot2 functions to reveal the frequency distribution of *Edge Type* field of *mc1\_edges\_raw*.

ggplot(data = mc1\_edges\_raw,  
 aes(y = `Edge Type`)) +  
 geom\_bar()



# 3 Data Pre-processing

## 3.1 Adding identifying columns

As a large part of this mini-challenge centers around Sailor Shift and the genre of “Oceanus Folk”, the following code will add columns to help with identification and filtering of Sailor Shift and the work in the genre of “Oceanus Folk”. This will help with analysis in addressing the questions and tasks.

mc1\_nodes\_raw <- mc1\_nodes\_raw %>%  
 mutate(  
 is\_sailor = (  
 str\_detect(name, regex("sailor shift", ignore\_case = TRUE))  
 ) %>% replace\_na(FALSE),  
   
 is\_oceanus\_folk = str\_detect(genre, regex("oceanus folk", ignore\_case = TRUE)) %>% #na/not oceanus folk = false  
 replace\_na(FALSE)  
 )

## 3.2 Converting date field

Date fields will be converted from *chr* to *int* for later analysis. Note that dates only appear for *Song* and *Album*.

mc1\_nodes\_raw <- mc1\_nodes\_raw %>%  
 mutate(across(c(release\_date, notoriety\_date, written\_date),  
 ~as.integer(if\_else(`Node Type` %in% c("Song", "Album"), ., NA\_character\_))))

## 3.3 Check for duplicates

### 3.3.1 Check for duplicates in *mc1\_nodes\_raw*

The following code chunk checks for *id* duplicates in *mc1\_nodes\_raw.*

mc1\_nodes\_raw %>%  
 count(id) %>%  
 filter(n > 1)

# A tibble: 0 × 2  
# ℹ 2 variables: id <int>, n <int>

There are no duplicated *id* in *mc1\_nodes\_raw*.

The following code checks for *name* duplicates in *mc1\_nodes\_raw*.

duplicated\_name <- mc1\_nodes\_raw %>%  
 count(name) %>%  
 filter(n > 1)  
  
duplicated\_name

# A tibble: 1,611 × 2  
 name n  
 <chr> <int>  
 1 Agata Records 2  
 2 Ancestral Echoes 2  
 3 Angela Thompson 2  
 4 Anthony Davis 2  
 5 Anthony Smith 2  
 6 Asuka Takahashi 3  
 7 Brandon Wilson 2  
 8 Brian Gonzalez 2  
 9 Bryan Garcia 2  
10 Bryan Smith 3  
# ℹ 1,601 more rows

The following code chunk shows all rows from *mc1\_nodes\_raw* that have duplicated names, and sorting them alphabetically by the *name* column. There are a total of 4,953 records with duplicated names in *mc1\_nodes\_raw*.

mc1\_nodes\_raw %>%  
 filter(name %in% duplicated\_name$name) %>%  
 arrange(name)

# A tibble: 4,953 × 12  
 `Node Type` name single release\_date genre notable id written\_date  
 <chr> <chr> <lgl> <int> <chr> <lgl> <int> <int>  
 1 RecordLabel Agata Recor… NA NA <NA> NA 1528 NA  
 2 RecordLabel Agata Recor… NA NA <NA> NA 17388 NA  
 3 Song Ancestral E… TRUE 1991 Drea… FALSE 11793 NA  
 4 Song Ancestral E… FALSE 2039 Avan… TRUE 17133 NA  
 5 Person Angela Thom… NA NA <NA> NA 1150 NA  
 6 Person Angela Thom… NA NA <NA> NA 13448 NA  
 7 Person Anthony Dav… NA NA <NA> NA 8692 NA  
 8 Person Anthony Dav… NA NA <NA> NA 12452 NA  
 9 Person Anthony Smi… NA NA <NA> NA 5719 NA  
10 Person Anthony Smi… NA NA <NA> NA 7694 NA  
# ℹ 4,943 more rows  
# ℹ 4 more variables: stage\_name <chr>, notoriety\_date <int>, is\_sailor <lgl>,  
# is\_oceanus\_folk <lgl>

### 3.3.2 Fixing duplicates in *mc1\_nodes\_raw*

The section will focus on fixing the duplicates found in *mc1\_nodes\_raw* as identified in section 3.3.1*.*

The following code chunk will tag each row with a unique key (*group\_key*) based on its respective column values. This helps to identify unique records.

# Step 1: Mark all node rows with a hash key for grouping  
mc1\_nodes\_tagged <- mc1\_nodes\_raw %>%  
 mutate(group\_key = paste(`Node Type`, name, single, release\_date, genre,  
 notable, written\_date, notoriety\_date, is\_sailor,  
 is\_oceanus\_folk, sep = "|"))  
  
mc1\_nodes\_tagged

# A tibble: 17,412 × 13  
 `Node Type` name single release\_date genre notable id written\_date  
 <chr> <chr> <lgl> <int> <chr> <lgl> <int> <int>  
 1 Song Breaking Th… TRUE 2017 Ocea… TRUE 0 NA  
 2 Person Carlos Duffy NA NA <NA> NA 1 NA  
 3 Person Min Qin NA NA <NA> NA 2 NA  
 4 Person Xiuying Xie NA NA <NA> NA 3 NA  
 5 RecordLabel Nautical Mi… NA NA <NA> NA 4 NA  
 6 Song Unshackled … FALSE 2026 Lo-F… TRUE 5 NA  
 7 Person Luke Payne NA NA <NA> NA 6 NA  
 8 Person Xiulan Zeng NA NA <NA> NA 7 NA  
 9 Person David Frank… NA NA <NA> NA 8 NA  
10 RecordLabel Colline-Cas… NA NA <NA> NA 9 NA  
# ℹ 17,402 more rows  
# ℹ 5 more variables: stage\_name <chr>, notoriety\_date <int>, is\_sailor <lgl>,  
# is\_oceanus\_folk <lgl>, group\_key <chr>

The code below deduplicates the dataset using *group\_key*, reducing the number of duplicated names from 4,953 to 14. The remaining 14 names appear more than once because their corresponding records differ in at least one column used to form *group\_key*, so they are retained as distinct entries.

# Step 2: Deduplicate and keep the preferred (with stage\_name if available)  
mc1\_nodes\_dedup <- mc1\_nodes\_tagged %>%  
 group\_by(group\_key) %>%  
 arrange(desc(!is.na(stage\_name))) %>%  
 slice(1) %>%  
 ungroup()  
  
duplicated\_name <- mc1\_nodes\_dedup %>%  
 count(name) %>%  
 filter(n > 1)  
  
mc1\_nodes\_raw %>%  
 filter(name %in% duplicated\_name$name) %>%  
 arrange(name)

# A tibble: 14 × 12  
 `Node Type` name single release\_date genre notable id written\_date  
 <chr> <chr> <lgl> <int> <chr> <lgl> <int> <int>  
 1 Song Ancestral E… TRUE 1991 Drea… FALSE 11793 NA  
 2 Song Ancestral E… FALSE 2039 Avan… TRUE 17133 NA  
 3 RecordLabel Coastal Ech… NA NA <NA> NA 4022 NA  
 4 Album Coastal Ech… NA 2023 Psyc… TRUE 15065 2019  
 5 Song Postcards f… TRUE 2023 Indi… TRUE 12852 2023  
 6 Song Postcards f… FALSE 1984 Acou… FALSE 17214 NA  
 7 Album Shattered R… NA 2013 Emo/… TRUE 3325 2013  
 8 Song Shattered R… FALSE 2036 Dark… TRUE 17088 NA  
 9 Song Unheard Fre… TRUE 2025 Alte… TRUE 7999 NA  
10 RecordLabel Unheard Fre… NA NA <NA> NA 10952 NA  
11 Song Vanishing P… TRUE 2018 Avan… TRUE 9371 2018  
12 Song Vanishing P… FALSE 2013 Ocea… FALSE 17338 NA  
13 RecordLabel Vertical Ho… NA NA <NA> NA 2453 NA  
14 Album Vertical Ho… NA 2017 Doom… TRUE 9262 NA  
# ℹ 4 more variables: stage\_name <chr>, notoriety\_date <int>, is\_sailor <lgl>,  
# is\_oceanus\_folk <lgl>

### 3.3.3 Check for duplicates in *mc1\_edges\_raw*

The following code proceeds to check for duplicates in *mc1\_edges\_raw.*

# Step 1: Identify duplicate combinations  
duplicate\_summary <- mc1\_edges\_raw %>%  
 count(source, target, `Edge Type`) %>%  
 filter(n > 1)  
  
# Step 2: Join back to get all original duplicate rows  
mc1\_edges\_raw %>%  
 inner\_join(duplicate\_summary, by = c("source", "target", "Edge Type"))

# A tibble: 6 × 5  
 `Edge Type` source target key n  
 <chr> <int> <int> <int> <int>  
1 PerformerOf 17057 17058 0 2  
2 PerformerOf 17057 17058 1 2  
3 PerformerOf 17349 17350 0 2  
4 PerformerOf 17349 17350 2 2  
5 PerformerOf 17355 17356 0 2  
6 PerformerOf 17355 17356 2 2

There are duplicates as seen above, with only differences in *key*. As *key* will not be used in subsequent data analysis, the duplicated edges will be removed with the following code.

mc1\_edges\_raw <- mc1\_edges\_raw %>%  
 distinct(source, target, `Edge Type`, .keep\_all = TRUE) %>%  
 select(!key)  
  
mc1\_edges\_raw %>%  
 count(source, target, `Edge Type`) %>%  
 filter(n > 1)

# A tibble: 0 × 4  
# ℹ 4 variables: source <int>, target <int>, Edge Type <chr>, n <int>

# 4 EDA

## 4.1 Explore and inspect Nodes

mc1\_nodes\_raw$release\_date %>% unique()

[1] 2017 NA 2026 2020 2027 2022 2007 2010 2003 2023 1997 2013 2000 2025 2029  
[16] 2015 2018 2016 2014 2028 2021 2030 2011 1994 2004 1998 1991 1999 2024 2012  
[31] 2002 2006 2008 2019 1995 1989 2032 2009 2001 1996 1990 1984 2005 1993 1986  
[46] 1985 1981 1992 1987 1988 1983 2031 1975 2035 2033 2037 2036 2039 2038 2034  
[61] 1977 1979 1980 1982 2040

mc1\_nodes\_raw %>%  
 filter(grepl("Sailor Shift", name)) #Sailor Shift is in name column and not in stage\_name column

# A tibble: 1 × 12  
 `Node Type` name single release\_date genre notable id written\_date  
 <chr> <chr> <lgl> <int> <chr> <lgl> <int> <int>  
1 Person Sailor Shift NA NA <NA> NA 17255 NA  
# ℹ 4 more variables: stage\_name <chr>, notoriety\_date <int>, is\_sailor <lgl>,  
# is\_oceanus\_folk <lgl>

' will be removed from *name* to prevent issues with tooltip in tidygraph.

mc1\_nodes\_clean <- mc1\_nodes\_raw %>%  
 mutate(  
 name = gsub("'", "", name))   
   
kable(head(mc1\_nodes\_clean))

| Node Type | name | single | release\_date | genre | notable | id | written\_date | stage\_name | notoriety\_date | is\_sailor | is\_oceanus\_folk |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Song | Breaking These Chains | TRUE | 2017 | Oceanus Folk | TRUE | 0 | NA | NA | NA | FALSE | TRUE |
| Person | Carlos Duffy | NA | NA | NA | NA | 1 | NA | NA | NA | FALSE | FALSE |
| Person | Min Qin | NA | NA | NA | NA | 2 | NA | NA | NA | FALSE | FALSE |
| Person | Xiuying Xie | NA | NA | NA | NA | 3 | NA | NA | NA | FALSE | FALSE |
| RecordLabel | Nautical Mile Records | NA | NA | NA | NA | 4 | NA | NA | NA | FALSE | FALSE |
| Song | Unshackled Heart | FALSE | 2026 | Lo-Fi Electronica | TRUE | 5 | NA | NA | NA | FALSE | FALSE |

## 4.2 Explore and inspect Edges

The following code chunk is used to ensure that *id* used in *mc1\_edges\_raw* matches the range range of *id* in *mc1\_nodes\_clean.*

range(mc1\_nodes\_clean$id)

[1] 0 17411

range(mc1\_edges\_raw$source)

[1] 0 17411

range(mc1\_edges\_raw$target)

[1] 0 17411

## 4.3 Mapping Node name to Edges id

tidygraph uses *from* and *to* columns to reference nodes. By default, tidygraph matches these edges reference against the first column in the nodes table, or against *name* column.

Currently, *source* and *target* columns in *mc1\_edges\_raw* contain *id* values that correspond to the *id* column in *mc1\_nodes\_clean*. To properly integrate with tidygraph’s conventions, the following will be done:

1. Restructure *mc1\_nodes\_clean*
   1. Rename the current *name* column to *node\_name* - this is done to preserve the actual node names
   2. Rename the *id* column to *name* so it becomes the primary identifier column that tidygraph will use for matching
2. Rename *source* and *target* columns in *mc1\_edges\_raw*, as required by tidygraph
3. Ensure data type consistency: Convert the *name* column (formerly *id*) to character format to match the data type of the edge references

mc1\_nodes\_clean <- mc1\_nodes\_dedup %>%  
 rename(node\_name = name, name = id) %>%  
 mutate(name = as.character(name)) %>%  
 select(`Node Type`, node\_name, release\_date, genre, notable, name, single, written\_date, stage\_name, notoriety\_date, is\_sailor, is\_oceanus\_folk)  
  
mc1\_nodes\_clean

# A tibble: 14,077 × 12  
 `Node Type` node\_name release\_date genre notable name single written\_date  
 <chr> <chr> <int> <chr> <lgl> <chr> <lgl> <int>  
 1 Album A Lush Dyst… 2031 Psyc… TRUE 17005 NA 2030  
 2 Album Addicted to… 2004 Sout… TRUE 14658 NA 2000  
 3 Album Adriatic Em… 2013 Post… TRUE 10412 NA NA  
 4 Album Aerial Echo… 2023 Indi… TRUE 7908 NA NA  
 5 Album Aftershock … 2028 Drea… TRUE 2030 NA 2021  
 6 Album Allegretto … 2020 Indi… TRUE 6251 NA NA  
 7 Album Alleys and … 2029 Jazz… TRUE 1310 NA NA  
 8 Album Alloy Archi… 2017 Indi… TRUE 8428 NA 2017  
 9 Album Almost (But… 2027 Alte… TRUE 14611 NA 2027  
10 Album Altar of De… 2020 Ocea… TRUE 5883 NA NA  
# ℹ 14,067 more rows  
# ℹ 4 more variables: stage\_name <chr>, notoriety\_date <int>, is\_sailor <lgl>,  
# is\_oceanus\_folk <lgl>

### 4.3.1 Creating edge mapping from old id to kept id

In section 3.3.2, duplicated nodes were deduplicated and removed, edges referring to the removed nodes will become invalid thus, edges will be remapped to the retained nodes. This ensures that all edges correctly point to existing nodes in the deduplicated graph.

# Step 1: Create mapping of all group\_key → kept id  
key\_to\_id\_map <- mc1\_nodes\_dedup %>%  
 select(group\_key, kept\_id = id)  
  
# Step 2: Map all original rows to the retained ID  
id\_remap <- mc1\_nodes\_tagged %>%  
 left\_join(key\_to\_id\_map, by = "group\_key") %>%  
 select(original\_id = id, kept\_id)  
  
id\_remap

# A tibble: 17,412 × 2  
 original\_id kept\_id  
 <int> <int>  
 1 0 0  
 2 1 1  
 3 2 14470  
 4 3 3  
 5 4 4  
 6 5 5  
 7 6 6  
 8 7 7  
 9 8 8  
10 9 9  
# ℹ 17,402 more rows

# Step 3: Replace edges' source and target with mapped kept\_id  
mc1\_edges\_mapped <- mc1\_edges\_raw %>%  
 left\_join(id\_remap, by = c("source" = "original\_id"))  
  
mc1\_edges\_mapped

# A tibble: 37,854 × 4  
 `Edge Type` source target kept\_id  
 <chr> <int> <int> <int>  
 1 InterpolatesFrom 0 1841 0  
 2 RecordedBy 0 4 0  
 3 PerformerOf 1 0 1  
 4 ComposerOf 1 16180 1  
 5 PerformerOf 2 0 14470  
 6 ProducerOf 2 16180 14470  
 7 PerformerOf 3 0 3  
 8 InterpolatesFrom 5 5088 5  
 9 InStyleOf 5 14332 5  
10 InterpolatesFrom 5 11677 5  
# ℹ 37,844 more rows

mc1\_edges\_mapped <- mc1\_edges\_mapped %>%  
 mutate(source = kept\_id) %>%  
 select(-kept\_id) %>%  
 left\_join(id\_remap, by = c("target" = "original\_id")) %>%  
 mutate(target = kept\_id) %>%  
 select(-kept\_id) %>%  
 rename(from = source, to = target) %>%  
 mutate(from = as.character(from), to = as.character(to))  
  
mc1\_edges\_mapped

# A tibble: 37,854 × 3  
 `Edge Type` from to   
 <chr> <chr> <chr>  
 1 InterpolatesFrom 0 1841   
 2 RecordedBy 0 4   
 3 PerformerOf 1 0   
 4 ComposerOf 1 16180  
 5 PerformerOf 14470 0   
 6 ProducerOf 14470 16180  
 7 PerformerOf 3 0   
 8 InterpolatesFrom 5 5088   
 9 InStyleOf 5 14332  
10 InterpolatesFrom 5 11677  
# ℹ 37,844 more rows

## 4.4 Remove unmatched edges

The following code chunk removes edges that reference missing node id, ensuring that only valid edges are kept.

mc1\_edges\_clean <- mc1\_edges\_mapped %>%  
 filter(!is.na(from), !is.na(to))

There are no unmatched edges.

## 4.5 Check for missing nodes

The following code chunk checks for missing nodes being referenced in *mc1\_edges\_clean* that do not exist in *mc1\_nodes\_clean*.

setdiff(  
 unique(c(mc1\_edges\_clean$from, mc1\_edges\_clean$to)),  
 mc1\_nodes\_clean$name  
)

character(0)

There are no missing nodes.

## 4.6 Validate Edges Schema

This section aims to ensure that each edge in the graph adheres to the schema specified in the VAST Challenge 2025 MC1 Data Description document. The following code checks whether the node types connect by each edge matches the valid source and target types for that edge’s type.

# Define valid source and destination types for each edge type  
edge\_rules <- list(  
 PerformerOf = list(source = c("Person", "MusicalGroup"), target = c("Song", "Album")),  
 ComposerOf = list(source = c("Person"), target = c("Song", "Album")),  
 ProducerOf = list(source = c("Person", "RecordLabel"), target = c("Song", "Album", "Person", "MusicalGroup")),  
 LyricistOf = list(source = c("Person"), target = c("Song", "Album")),  
 RecordedBy = list(source = c("Song", "Album"), target = c("RecordLabel")),  
 DistributedBy = list(source = c("Song", "Album"), target = c("RecordLabel")),  
 InStyleOf = list(source = c("Song", "Album"), target = c("Song", "Album", "Person", "MusicalGroup")),  
 InterpolatesFrom = list(source = c("Song", "Album"), target = c("Song", "Album")),  
 CoverOf = list(source = c("Song", "Album"), target = c("Song", "Album")),  
 LyricalReferenceTo = list(source = c("Song", "Album"), target = c("Song", "Album")),  
 DirectlySamples = list(source = c("Song", "Album"), target = c("Song", "Album")),  
 MemberOf = list(source = c("Person"), target = c("MusicalGroup"))  
)

The following code chunk checks for any erroneous edge and node relationships defined in the code chunk above.

# Create a lookup for node types  
node\_type\_lookup <- mc1\_nodes\_clean %>%  
 select(name, `Node Type`) %>%  
 deframe()  
  
# Add source and target node types to the edge table  
mc1\_edges\_checked <- mc1\_edges\_clean %>%  
 mutate(  
 source\_type = node\_type\_lookup[from],  
 target\_type = node\_type\_lookup[to]  
 )  
  
mc1\_edges\_tagged <- mc1\_edges\_checked %>%  
 rowwise() %>%  
 mutate(  
 valid = {  
 rule <- edge\_rules[[`Edge Type`]]  
 if (is.null(rule)) TRUE  
 else {  
 source\_type %in% rule$source && target\_type %in% rule$target  
 }  
 }  
 ) %>%  
 ungroup()  
  
# Count and display invalid edge combinations  
invalid\_edge\_summary <- mc1\_edges\_tagged %>%  
 filter(!valid) %>%  
 count(`Edge Type`, source\_type, target\_type, sort = TRUE)  
  
print(invalid\_edge\_summary)

# A tibble: 24 × 4  
 `Edge Type` source\_type target\_type n  
 <chr> <chr> <chr> <int>  
 1 LyricistOf MusicalGroup Song 106  
 2 RecordedBy RecordLabel Album 102  
 3 ProducerOf MusicalGroup Song 100  
 4 ComposerOf MusicalGroup Song 97  
 5 ProducerOf MusicalGroup Album 31  
 6 LyricistOf MusicalGroup Album 28  
 7 ComposerOf MusicalGroup Album 17  
 8 InStyleOf MusicalGroup MusicalGroup 12  
 9 InStyleOf Person MusicalGroup 11  
10 InterpolatesFrom MusicalGroup MusicalGroup 10  
# ℹ 14 more rows

# Check total invalid edge count  
cat("Total invalid edges:", sum(!mc1\_edges\_tagged$valid), "\n")

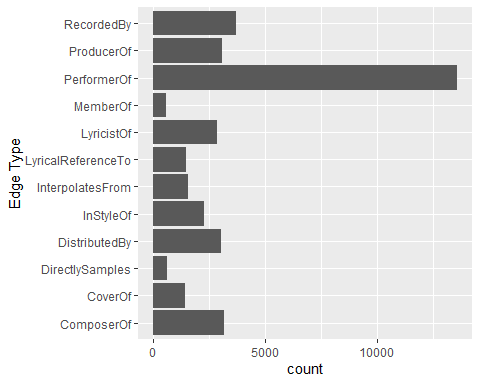
Total invalid edges: 550

There are 550 edges that do not adhere to the schema specified in the data description file provided. The following code will remove these edges.

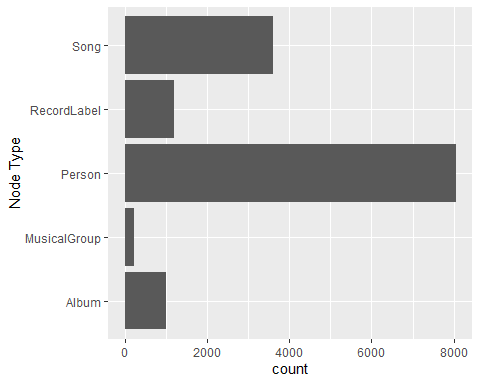
# Keep only valid edges  
mc1\_edges\_clean <- mc1\_edges\_tagged %>%  
 filter(valid) %>%  
 select(from, to, `Edge Type`) # drop helper columns

## 4.7 Visualising Edge and Node types

ggplot(data = mc1\_edges\_clean,  
 aes(y = `Edge Type`)) +  
 geom\_bar()



ggplot(data = mc1\_nodes\_clean,  
 aes(y = `Node Type`)) +  
 geom\_bar()



## 4.8 Creating knowledge graph

tbl\_graph() is used to create tidygraph’s graph object by using the following code chunk.

music = tbl\_graph(edges = mc1\_edges\_clean,  
 nodes = mc1\_nodes\_clean,  
 directed = TRUE)  
  
class(music)

[1] "tbl\_graph" "igraph"

Several of the ggraph layouts involve randomisation. In order to ensure reproducibility, it is necessary to set the seed value before plotting by using the code chunk below.

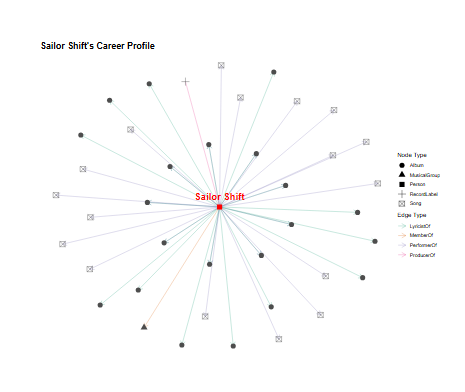
set.seed(1234)

# 5 VAST Challenge 2025 Mini-Challenge 1

For Task 1, it is to design and develop visualizations and visual analytic tools that will allow Silas to explore and understand the profile of Sailor Shift’s career. We start off with a network visualisation to provide an overview of Sailor Shift’s works throughout her career, as well as the various roles she played in these works.

## Network Visualisation

# Step 1: Identify Sailor Shift using the is\_sailor column  
sailor\_vertex\_name <- mc1\_nodes\_clean %>%  
 filter(is\_sailor) %>%  
 pull(name) %>%  
 first()  
  
# Step 2: Prepare edges and nodes related to Sailor Shift  
sailor\_edges <- mc1\_edges\_clean %>%  
 filter(from == sailor\_vertex\_name | to == sailor\_vertex\_name)  
  
sailor\_node\_names <- unique(c(sailor\_edges$from, sailor\_edges$to))  
  
sailor\_nodes <- mc1\_nodes\_clean %>%  
 filter(name %in% sailor\_node\_names) %>%  
 distinct(name, .keep\_all = TRUE)  
  
# Step 3: Build tbl\_graph object and annotate nodes  
career\_graph <- tbl\_graph(nodes = sailor\_nodes, edges = sailor\_edges, directed = TRUE) %>%  
 activate(nodes) %>%  
 mutate(  
 node\_color = ifelse(is\_sailor, "red", "grey30"),  
 tooltip\_text = paste0(  
 "Name: ", node\_name, "\n",  
 "Type: ", `Node Type`, "\n",  
 ifelse(!is.na(genre), paste0("Genre: ", genre, "\n"), ""),  
 ifelse(!is.na(release\_date), paste0("Release: ", release\_date, "\n"), "")  
 )  
 )  
  
# Step 4: Extract layout coordinates  
layout\_df <- create\_layout(career\_graph, layout = "fr") %>%  
 as\_tibble() %>%  
 select(name, x, y)  
  
nodes\_plot <- career\_graph %>%  
 as\_tibble() %>%  
 left\_join(layout\_df, by = "name")  
  
edges\_plot <- sailor\_edges %>%  
 left\_join(nodes\_plot %>% select(name, x, y), by = c("from" = "name")) %>%  
 rename(x\_from = x, y\_from = y) %>%  
 left\_join(nodes\_plot %>% select(name, x, y), by = c("to" = "name")) %>%  
 rename(x\_to = x, y\_to = y)  
  
# Get Sailor Shift node coordinates for annotation  
sailor\_coords <- nodes\_plot %>%  
 filter(is\_sailor) %>%  
 select(x, y)  
  
# Step 5: Plot with ggiraph  
p <- ggplot() +  
 # Edge layer with its own color scale  
 geom\_segment(  
 data = edges\_plot,  
 aes(  
 x = x\_from, y = y\_from, xend = x\_to, yend = y\_to,  
 color = `Edge Type`  
 ),  
 alpha = 0.4, arrow = arrow(length = unit(3, 'mm'))  
 ) +  
 scale\_color\_brewer(palette = "Dark2", name = "Edge Type") +  
  
 # New color scale for nodes  
 ggnewscale::new\_scale\_color() +  
  
 geom\_point\_interactive(  
 data = nodes\_plot,  
 aes(  
 x = x, y = y,  
 tooltip = tooltip\_text,  
 data\_id = name,  
 color = node\_color,  
 shape = `Node Type`  
 ),  
 size = 4  
 ) +  
 scale\_color\_manual(  
 values = c("red" = "red", "grey30" = "grey30")  
 ) +  
 guides(color = "none") + # Remove node color legend  
  
 # Add Sailor Shift label in the middle of the graph  
 geom\_text(  
 data = sailor\_coords,  
 aes(x = x, y = y, label = "Sailor Shift"),  
 size = 6, fontface = "bold", color = "red", vjust = -1  
 ) +  
  
 theme\_void() +  
 labs(title = "Sailor Shift's Career Profile") +  
 theme(  
 plot.title = element\_text(size = 16, face = "bold")  
 )  
  
girafe(ggobj = p, width\_svg = 10, height\_svg = 8)



## Role Summary

# Filter edges related to Sailor Shift  
sailor\_edges <- mc1\_edges\_clean %>%  
 filter(from == sailor\_vertex\_name | to == sailor\_vertex\_name)  
  
# Count edges by Edge Type  
edge\_counts <- sailor\_edges %>%  
 count(`Edge Type`) %>%  
 arrange(desc(n))  
  
# Display as a simple table  
kable(edge\_counts, col.names = c("Role (Edge Type)", "Count"), caption = "Sailor Shift's Career Roles")

Sailor Shift’s Career Roles

| Role (Edge Type) | Count |
| --- | --- |
| PerformerOf | 26 |
| LyricistOf | 21 |
| MemberOf | 1 |
| ProducerOf | 1 |

## Summary of works

# Filter edges related to Sailor Shift  
sailor\_edges <- mc1\_edges\_clean %>%  
 filter(from == sailor\_vertex\_name | to == sailor\_vertex\_name)  
  
# Get the names of nodes connected to Sailor Shift (excluding Sailor Shift themself)  
related\_names <- unique(c(sailor\_edges$from, sailor\_edges$to))  
related\_names <- setdiff(related\_names, sailor\_vertex\_name)  
  
# Get node types for these connected nodes  
related\_nodes <- mc1\_nodes\_clean %>%  
 filter(name %in% related\_names)  
  
# Count by Node Type (Song, Album)  
type\_counts <- related\_nodes %>%  
 filter(`Node Type` %in% c("Song", "Album")) %>%  
 count(`Node Type`) %>%  
 arrange(desc(n))  
  
# Display as a simple table  
kable(type\_counts, col.names = c("Type", "Count"), caption = "Number of Songs and Albums Related to Sailor Shift")

Number of Songs and Albums Related to Sailor Shift

| Type | Count |
| --- | --- |
| Album | 21 |
| Song | 17 |

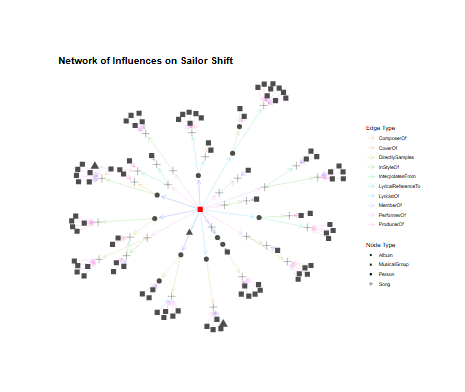
Based on the above, Sailor Shift is primarily a performer and lyricist, and a member of Ivy Echos, a musical group. She was a lyricist for 17 songs and performer of 21 albums. It also shows that Oceanic Records, a Record label, has participated in the production of her works.

## 5.1 Question 1a - Who has she been most influenced by over time?

The network structure below shows how Sailor Shift’s career has been influenced by others. PageRank is used to measure the overall influence of each person, musical group or work within the network. This captures both direct and indirect influences.

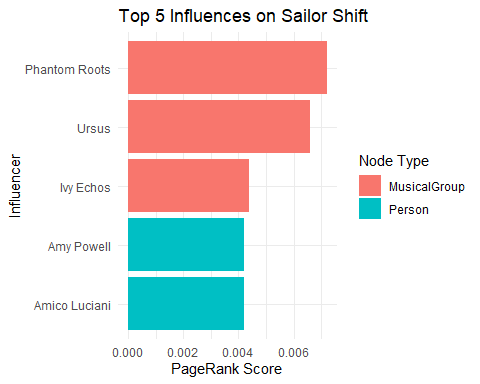
## Network Visualisation

# Step 0: Get name of 'Sailor Shift'  
sailor\_vertex\_name <- mc1\_nodes\_clean %>%  
 filter(is\_sailor == TRUE) %>%  
 pull(name) %>%  
 first()  
  
# Step 1: Find direct influence relationships from Sailor Shift  
# These are the artists/works that Sailor Shift has been influenced by  
direct\_influence\_types <- c("InStyleOf", "CoverOf", "InterpolatesFrom", "LyricalReferenceTo", "DirectlySamples")  
  
sailor\_direct\_influences <- mc1\_edges\_clean %>%  
 filter(from == sailor\_vertex\_name,   
 `Edge Type` %in% direct\_influence\_types)  
  
# Step 2: Get immediate neighbors (people/groups Sailor Shift works with)  
sailor\_out\_edges <- mc1\_edges\_clean %>%  
 filter(from == sailor\_vertex\_name)  
  
sailor\_out\_node\_names <- sailor\_out\_edges$to  
  
# Step 3: Split into people/groups vs songs/albums  
sailor\_person\_group <- mc1\_nodes\_clean %>%  
 filter(name %in% sailor\_out\_node\_names, `Node Type` %in% c("Person", "MusicalGroup")) %>%  
 pull(name)  
  
sailor\_songs\_all <- mc1\_nodes\_clean %>%  
 filter(name %in% sailor\_out\_node\_names, `Node Type` %in% c("Song", "Album")) %>%  
 pull(name)  
  
# Step 4: For songs/albums, find their direct influences too  
song\_influences <- mc1\_edges\_clean %>%  
 filter(from %in% sailor\_songs\_all,   
 `Edge Type` %in% direct\_influence\_types)  
  
# Step 5: Get all influence targets (who influenced Sailor Shift or their works)  
all\_influence\_targets <- unique(c(  
 sailor\_direct\_influences$to,  
 song\_influences$to  
))  
  
# Step 6: Get creators of Sailor Shift's works (indirect influence indicators)  
creator\_edge\_types <- c("PerformerOf", "ComposerOf", "ProducerOf", "LyricistOf")  
  
sailor\_songs <- mc1\_edges\_clean %>%  
 filter(from %in% sailor\_songs\_all) %>%  
 pull(from) %>%  
 unique()  
  
sailor\_songs\_out\_nodes <- mc1\_edges\_clean %>%  
 filter(from %in% sailor\_songs) %>%  
 pull(to)  
  
creator\_edges <- mc1\_edges\_clean %>%  
 filter(to %in% sailor\_songs\_out\_nodes, `Edge Type` %in% creator\_edge\_types)  
  
sailor\_people\_group\_neighbourhood\_nodes <- creator\_edges %>%  
 pull(from) %>%  
 unique()  
  
# Step 7: Combine all relevant nodes for subgraph  
sailor\_all\_node\_names <- unique(c(  
 sailor\_vertex\_name,  
 sailor\_person\_group,  
 sailor\_songs,  
 sailor\_songs\_out\_nodes,  
 sailor\_people\_group\_neighbourhood\_nodes,  
 all\_influence\_targets   
))  
  
# Step 8: Create subgraph  
sub\_music <- music %>%  
 filter(name %in% sailor\_all\_node\_names)  
  
# Step 9: Calculate PageRank   
sub\_music <- sub\_music %>%  
 activate(nodes) %>%  
 mutate(  
 pagerank = centrality\_pagerank()  
 )  
  
# Step 10: Set node size based on PageRank for people/groups, fixed for others  
sub\_music <- sub\_music %>%  
 mutate(  
 is\_sailor = name == sailor\_vertex\_name,  
 node\_color = ifelse(is\_sailor, "red", "grey30"),  
 tooltip\_text = sprintf(  
 "Name: %s\nType: %s\nPageRank: %.4f",  
 node\_name, `Node Type`, pagerank  
 ),  
 node\_size = case\_when(  
 `Node Type` %in% c("Person", "MusicalGroup") ~ rescale(pagerank, to = c(4, 20)),  
 TRUE ~ 4   
 )  
 )  
  
# Step 11: Create visualization  
g <- sub\_music %>%  
 ggraph(layout = "fr") +  
 geom\_edge\_link(  
 aes(color = `Edge Type`),   
 alpha = 0.3,  
 arrow = arrow(length = unit(3, 'mm')),  
 end\_cap = circle(3, 'mm')  
 ) +  
 geom\_point\_interactive(  
 aes(  
 x = x, y = y,  
 data\_id = name,  
 tooltip = tooltip\_text,  
 shape = `Node Type`,  
 colour = node\_color,  
 size = node\_size  
 )  
 ) +  
 scale\_shape\_discrete(name = "Node Type") +  
 scale\_colour\_identity() +  
 scale\_size\_identity() +  
 theme\_graph(base\_family = "sans") +  
 labs(  
 title = "Network of Influences on Sailor Shift"  
 )  
  
girafe(ggobj = g, width\_svg = 10, height\_svg = 8)



## Influence Summary

# Filter to people and groups only, exclude Sailor Shift node itself  
top\_influencers <- sub\_music %>%  
 as\_tibble() %>%  
 filter(  
 `Node Type` %in% c("Person", "MusicalGroup"),  
 name != sailor\_vertex\_name  
 ) %>%  
 arrange(desc(pagerank)) %>%  
 slice\_head(n = 5)  
  
# Plot  
ggplot(top\_influencers, aes(x = reorder(node\_name, pagerank), y = pagerank, fill = `Node Type`)) +  
 geom\_bar(stat = "identity") +  
 coord\_flip() +  
 labs(  
 title = "Top 5 Influences on Sailor Shift",  
 x = "Influencer",  
 y = "PageRank Score"  
 ) +  
 theme\_minimal()

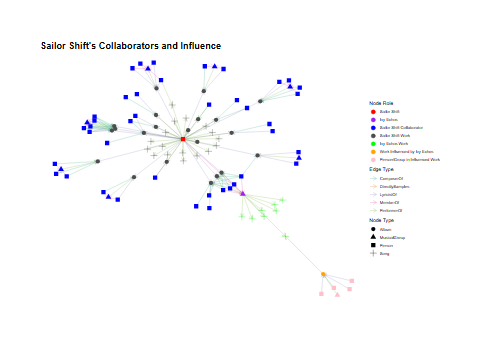


Based on the PageRank score, it is noted that she is most influenced by musical groups as the top 3 most influences are musical groups. Phantom Roots have influenced her the most over time, this is followed by Ursus and the group she was a part of, Ivy Echos.

## 5.2 Question 1b - Who has she collaborated with and directly or indirectly influenced?

The network visualisation below explores Sailor Shift’s collaborations and influence. While the primary question centers on Sailor Shift, the analysis also incorporates Ivy Echos, the musical group that she was a member of. Including Ivy Echos is essential because Sailor Shift’s creative impact can extend beyond her solo work as her contributions as part of Ivy Echos could have influenced others. The visualisation therefore highlights not just individuals and groups who have collaborated with Sailor Shift on her works, but also those influenced by Ivy Echos, providing an extensive picture of her influence.

# Step 1: Define all relevant edge types per schema  
collab\_credit\_types <- c("PerformerOf", "ComposerOf", "ProducerOf", "LyricistOf", "MemberOf")  
influence\_types <- c("CoverOf", "InterpolatesFrom", "LyricalReferenceTo", "DirectlySamples", "InStyleOf")  
  
# Step 2: Get Sailor Shift's node ID  
sailor\_vertex\_name <- mc1\_nodes\_clean %>%  
 filter(is\_sailor == TRUE) %>%  
 pull(name) %>% first()  
  
# Step 3: Find all Sailor Shift's works (songs/albums she performed or was lyricist of)  
sailor\_works <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% c("PerformerOf", "LyricistOf"), from == sailor\_vertex\_name) %>%  
 pull(to)  
  
# Step 4: Find all Person/MusicalGroup collaborated on Sailor Shift's works (excluding herself)  
sailor\_collab\_edges <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% collab\_credit\_types, to %in% sailor\_works, from != sailor\_vertex\_name)  
sailor\_collab\_nodes <- mc1\_nodes\_clean %>%  
 filter(name %in% sailor\_collab\_edges$from, `Node Type` %in% c("Person", "MusicalGroup")) %>%  
 pull(name)  
  
# Step 5: Get Ivy Echos's node ID and works  
ivy\_echos\_name <- mc1\_nodes\_clean %>%  
 filter(str\_detect(node\_name, regex("Ivy Echos", ignore\_case = TRUE))) %>%  
 pull(name) %>% first()  
ivy\_works <- mc1\_edges\_clean %>%  
 filter(`Edge Type` == "PerformerOf", from == ivy\_echos\_name) %>%  
 pull(to)  
ivy\_works <- mc1\_nodes\_clean %>%  
 filter(name %in% ivy\_works, `Node Type` %in% c("Song", "Album")) %>%  
 pull(name)  
  
# Step 6: Find all works influenced by Ivy Echos's works (Ivy Echos's works as destination of influence edges)  
ivy\_influenced\_edges <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% influence\_types, to %in% ivy\_works)  
ivy\_influenced\_works <- ivy\_influenced\_edges$from  
  
# Step 7: For each influenced work, get the people/groups involved (collaborators on those works)  
ivy\_influenced\_collab\_edges <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% collab\_credit\_types, to %in% ivy\_influenced\_works)  
ivy\_influenced\_collab\_nodes <- mc1\_nodes\_clean %>%  
 filter(name %in% ivy\_influenced\_collab\_edges$from, `Node Type` %in% c("Person", "MusicalGroup")) %>%  
 pull(name)  
  
# Step 8: Collect all relevant nodes and edges for the network  
all\_relevant\_nodes <- unique(c(  
 sailor\_vertex\_name,  
 sailor\_collab\_nodes,  
 sailor\_works,  
 ivy\_echos\_name,  
 ivy\_works,  
 ivy\_influenced\_works,  
 ivy\_influenced\_collab\_nodes  
))  
  
all\_relevant\_edges <- mc1\_edges\_clean %>%  
 filter(from %in% all\_relevant\_nodes & to %in% all\_relevant\_nodes)  
  
# Step 9: Annotate node roles for plotting  
sub\_nodes\_df <- mc1\_nodes\_clean %>%  
 filter(name %in% all\_relevant\_nodes) %>%  
 mutate(  
 node\_role = case\_when(  
 name == sailor\_vertex\_name ~ "Sailor Shift",  
 name == ivy\_echos\_name ~ "Ivy Echos",  
 name %in% sailor\_collab\_nodes ~ "Sailor Shift Collaborator",  
 name %in% sailor\_works ~ "Sailor Shift Work",  
 name %in% ivy\_works ~ "Ivy Echos Work",  
 name %in% ivy\_influenced\_works ~ "Work Influenced by Ivy Echos",  
 name %in% ivy\_influenced\_collab\_nodes ~ "Person/Group in Influenced Work",  
 TRUE ~ "Other"  
 ),  
 node\_color = case\_when(  
 node\_role == "Sailor Shift" ~ "red",  
 node\_role == "Ivy Echos" ~ "purple",  
 node\_role == "Sailor Shift Collaborator" ~ "blue",  
 node\_role == "Sailor Shift Work" ~ "grey30",  
 node\_role == "Ivy Echos Work" ~ "green",  
 node\_role == "Work Influenced by Ivy Echos" ~ "orange",  
 node\_role == "Person/Group in Influenced Work" ~ "pink",  
 TRUE ~ "steelblue"  
 ),  
 tooltip\_text = paste0(  
 "Name: ", node\_name, "\n",  
 "Type: ", `Node Type`, "\n",  
 "Role: ", node\_role, "\n",  
 ifelse(!is.na(genre), paste0("Genre: ", genre, "\n"), ""),  
 ifelse(!is.na(release\_date), paste0("Release: ", release\_date, "\n"), "")  
 )  
 )  
  
# Step 10: Create tidygraph object and layout  
career\_graph <- tbl\_graph(nodes = sub\_nodes\_df, edges = all\_relevant\_edges, directed = TRUE) %>%  
 activate(nodes)  
  
layout\_df <- create\_layout(career\_graph, layout = "fr") %>%  
 as\_tibble() %>%  
 select(name, x, y)  
  
nodes\_plot <- as\_tibble(career\_graph) %>%  
 left\_join(layout\_df, by = "name")  
  
edges\_plot <- all\_relevant\_edges %>%  
 left\_join(nodes\_plot %>% select(name, x, y), by = c("from" = "name")) %>%  
 rename(x\_from = x, y\_from = y) %>%  
 left\_join(nodes\_plot %>% select(name, x, y), by = c("to" = "name")) %>%  
 rename(x\_to = x, y\_to = y)  
  
# Step 11: Get coordinates for annotation  
sailor\_coords <- nodes\_plot %>%  
 filter(name == sailor\_vertex\_name) %>%  
 select(x, y)  
ivy\_coords <- nodes\_plot %>%  
 filter(name == ivy\_echos\_name) %>%  
 select(x, y)  
  
# Step 12: Plot with ggplot2 + ggiraph, with annotation and legend  
p <- ggplot() +  
 geom\_segment(  
 data = edges\_plot,  
 aes(  
 x = x\_from, y = y\_from, xend = x\_to, yend = y\_to,  
 color = `Edge Type`  
 ),  
 alpha = 0.4, arrow = arrow(length = unit(3, 'mm'))  
 ) +  
 scale\_color\_brewer(palette = "Dark2", name = "Edge Type") +  
 ggnewscale::new\_scale\_color() +  
 geom\_point\_interactive(  
 data = nodes\_plot,  
 aes(  
 x = x, y = y,  
 tooltip = tooltip\_text,  
 data\_id = name,  
 color = node\_role,   
 shape = `Node Type`  
 ),  
 size = 4  
 ) +  
 scale\_color\_manual(  
 name = "Node Role",  
 values = c(  
 "Sailor Shift" = "red",  
 "Ivy Echos" = "purple",   
 "Sailor Shift Collaborator" = "blue",  
 "Sailor Shift Work" = "grey30",  
 "Ivy Echos Work" = "green",  
 "Work Influenced by Ivy Echos" = "orange",  
 "Person/Group in Influenced Work" = "pink",  
 "Other" = "steelblue"  
 ),  
 breaks = c(  
 "Sailor Shift",  
 "Ivy Echos",   
 "Sailor Shift Collaborator",  
 "Sailor Shift Work",  
 "Ivy Echos Work",  
 "Work Influenced by Ivy Echos",  
 "Person/Group in Influenced Work"  
 )  
 ) +  
 theme\_void() +  
 labs(title = "Sailor Shift's Collaborators and Influence") +  
 guides(  
 color = guide\_legend(  
 title = "Node Role",  
 override.aes = list(size = 4),  
 title.position = "top"  
 ),  
 shape = guide\_legend(  
 title = "Node Type",  
 title.position = "top"  
 )  
 ) +  
 theme(  
 legend.position = "right",  
 legend.box = "vertical",  
 plot.title = element\_text(size = 20, face = "bold")   
 )  
  
girafe(ggobj = p, width\_svg = 12, height\_svg = 8)



The visualisation shows a wide array of individuals and musical groups who have collaborated with Sailor Shift on various works, this reflects her active engagement within the industry. While there are no instances of Sailor Shift directly influencing other artists, the visualisation reveals that her group, Ivy Echos, has influenced a group and four individuals through a song (Deepsea Fireflies, released in 2025). This demonstrates that Sailor Shift’s reach extends beyond her personal collaborations, contributing to a broader legacy through her involvement with Ivy Echos.

## 5.3 Question 1c - How has she influenced collaborators of the broader Oceanus Folk community?

The network visualisation aims to analyse how Sailor Shift influenced collaborators of the broader Oceanus Folk community.

Sailor Shift and her group (Ivy Echos) were primary entities of interest, all works associated to them are compiled to form the foundation of Sailor Shift’s musical output. Based on this, several types of influence were analysed:

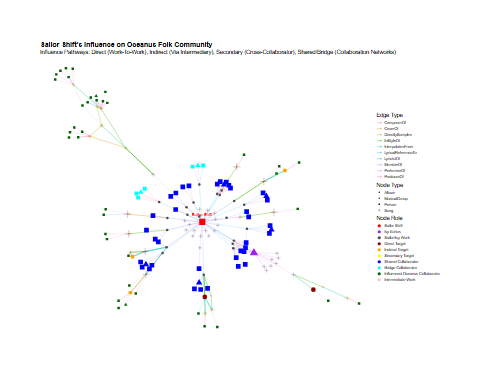
1. Direct influence - This includes Oceanus Folk collaborators’ works that were explicity influenced by Sailor Shift or Ivy Echos through relationships such as *CoverOf, InterpolatesFrom, LyricalReferenceTo, DirectlySamples,* and *InStyleOf*.
2. Indirect (two-step influence) - This occurs when a work by Sailor Shift or Ivy Echos influences an intermediate piece, which then goes on to influence a work by an Oceanus Folk collaborator. These two-step chains shows how Sailor Shift’s influence can propagate through the network.
3. Cross-collaborator influence - This captures intra-community influence where Oceanus Folk works that were initially influenced by Sailor Shift/Ivy Echos proceeded to influence other Oceanus Folk creations.
4. Collaboration-mediated influence - This is transmitted through shared or bridge collaborators.
   * Shared collaborators are individuals or groups who worked with both Sailor Shift/Ivy Echos and the Oceanus Folk community
   * Bridge Collaborators are those who first worked with Sailor Shift/Ivy Echos and later collaborated with Ocean Folk Contributors.

Based on the influences above, it reveals the full extent of Sailor Shift’s reach within the Oceanus Folk Community.

# Step 1: Define edge types  
collab\_credit\_types <- c("PerformerOf", "ComposerOf", "ProducerOf", "LyricistOf", "MemberOf")  
influence\_edge\_types <- c("CoverOf", "InterpolatesFrom", "LyricalReferenceTo", "DirectlySamples", "InStyleOf")  
  
# Step 2: Identify all nodes with genre == "Oceanus Folk"  
oceanus\_folk\_works <- mc1\_nodes\_clean %>%  
 filter(genre == "Oceanus Folk") %>%  
 pull(name)  
  
# Step 3: Identify all Person and MusicalGroup who are collaborators on Oceanus Folk works  
oceanus\_folk\_collaborators <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% collab\_credit\_types,   
 to %in% oceanus\_folk\_works) %>%  
 inner\_join(mc1\_nodes\_clean %>% select(name, `Node Type`), by = c("from" = "name")) %>%  
 filter(`Node Type` %in% c("Person", "MusicalGroup")) %>%  
 pull(from) %>%  
 unique()  
  
# Step 4: Get Sailor Shift and Ivy Echos  
sailor\_vertex\_name <- mc1\_nodes\_clean %>%  
 filter(is\_sailor == TRUE) %>%  
 pull(name) %>%   
 first()  
  
ivy\_echos\_name <- mc1\_edges\_clean %>%  
 filter(`Edge Type` == "MemberOf", from == sailor\_vertex\_name) %>%  
 pull(to) %>%  
 first()  
  
# Step 5: Find all works that Sailor Shift and Ivy Echos have created/performed  
sailor\_works <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% collab\_credit\_types, from == sailor\_vertex\_name) %>%  
 pull(to)  
  
ivy\_works <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% collab\_credit\_types, from == ivy\_echos\_name) %>%  
 pull(to)  
  
sailor\_ivy\_works <- unique(c(sailor\_works, ivy\_works))  
  
# Step 6: Find all works that the Oceanus Folk collaborators have worked on  
oceanus\_collaborator\_works <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% collab\_credit\_types,   
 from %in% oceanus\_folk\_collaborators) %>%  
 pull(to) %>%  
 unique()  
  
# Step 7: Direct influence - Sailor Shift/Ivy Echos works influencing Oceanus collaborator works  
direct\_influence <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% influence\_edge\_types,  
 from %in% sailor\_ivy\_works,  
 to %in% oceanus\_collaborator\_works) %>%  
 mutate(influence\_direction = "Sailor/Ivy → Oceanus",  
 pathway\_type = "Direct")  
  
# Step 8: Indirect influence - Multi-step pathways  
  
# 8a: Find intermediate works that could bridge Sailor Shift/Ivy Echos to Oceanus  
# Works influenced BY Sailor/Ivy  
sailor\_influenced\_works <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% influence\_edge\_types,  
 from %in% sailor\_ivy\_works) %>%  
 pull(to) %>%  
 unique()  
  
# Works that influence Sailor/Ivy   
sailor\_influencing\_works <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% influence\_edge\_types,  
 to %in% sailor\_ivy\_works) %>%  
 pull(from) %>%  
 unique()  
  
# All intermediate works in potential pathways  
intermediate\_works <- unique(c(sailor\_influenced\_works, sailor\_influencing\_works))  
  
# 8b: Two-step influence: Sailor/Ivy → Intermediate → Oceanus collaborators  
indirect\_influence\_step1 <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% influence\_edge\_types,  
 from %in% sailor\_ivy\_works,  
 to %in% intermediate\_works) %>%  
 select(sailor\_work = from, intermediate\_work = to, step1\_edge\_type = `Edge Type`)  
  
indirect\_influence\_step2 <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% influence\_edge\_types,  
 from %in% intermediate\_works,  
 to %in% oceanus\_collaborator\_works) %>%  
 select(intermediate\_work = from, oceanus\_work = to, step2\_edge\_type = `Edge Type`)  
  
# Join to find complete 2-step pathways  
two\_step\_pathways <- indirect\_influence\_step1 %>%  
 inner\_join(indirect\_influence\_step2, by = "intermediate\_work") %>%  
 mutate(pathway\_type = "Indirect (2-step)",  
 influence\_direction = "Sailor/Ivy → Intermediate → Oceanus")  
  
# 8c: Cross-collaborator influence within Oceanus community  
# Find Oceanus works that were influenced by Sailor and then influenced other Oceanus works  
directly\_influenced\_oceanus\_works <- unique(c(direct\_influence$to, two\_step\_pathways$oceanus\_work))  
  
cross\_collab\_influence <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% influence\_edge\_types,  
 from %in% directly\_influenced\_oceanus\_works,  
 to %in% oceanus\_collaborator\_works,  
 from != to) %>%  
 mutate(pathway\_type = "Cross-collaborator",  
 influence\_direction = "Sailor-influenced Oceanus work → Other Oceanus work")  
  
# Step 9: Collaboration-mediated influence   
  
# 9a: People who worked with both Sailor/Ivy AND Oceanus Folk collaborators  
sailor\_ivy\_collaborators <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% collab\_credit\_types,  
 to %in% sailor\_ivy\_works) %>%  
 inner\_join(mc1\_nodes\_clean %>% select(name, `Node Type`), by = c("from" = "name")) %>%  
 filter(`Node Type` %in% c("Person", "MusicalGroup")) %>%  
 pull(from) %>%  
 unique()  
  
shared\_collaborators <- intersect(sailor\_ivy\_collaborators, oceanus\_folk\_collaborators)  
  
# 9b. Bridge collaborators - worked with Sailor/Ivy, then later with other Oceanus Folk collaborators  
bridge\_collaborators <- setdiff(sailor\_ivy\_collaborators, oceanus\_folk\_collaborators)  
bridge\_to\_oceanus <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% collab\_credit\_types,  
 from %in% bridge\_collaborators) %>%  
 inner\_join(  
 mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% collab\_credit\_types,  
 from %in% oceanus\_folk\_collaborators) %>%  
 select(shared\_work = to),  
 by = c("to" = "shared\_work")  
 ) %>%  
 select(bridge\_person = from, shared\_work = to) %>%  
 distinct()  
  
# Step 10: Identify influenced Oceanus Folk Collaborators   
  
# Get all works that show influence from Sailor/Ivy  
all\_influenced\_oceanus\_works <- unique(c(  
 direct\_influence$to,  
 two\_step\_pathways$oceanus\_work,  
 cross\_collab\_influence$to  
))  
  
# Find which Oceanus Folk collaborators worked on these influenced works  
directly\_influenced\_collaborators <- mc1\_edges\_clean %>%  
 filter(`Edge Type` %in% collab\_credit\_types,   
 to %in% all\_influenced\_oceanus\_works) %>%  
 inner\_join(mc1\_nodes\_clean %>% select(name, `Node Type`), by = c("from" = "name")) %>%  
 filter(`Node Type` %in% c("Person", "MusicalGroup"),  
 from %in% oceanus\_folk\_collaborators) %>%  
 pull(from) %>%  
 unique()  
  
# Add collaborators connected through shared/bridge relationships  
collaboration\_influenced <- unique(c(shared\_collaborators, bridge\_to\_oceanus$bridge\_person))  
collaboration\_influenced <- intersect(collaboration\_influenced, oceanus\_folk\_collaborators)  
  
total\_influenced\_collaborators <- unique(c(directly\_influenced\_collaborators, collaboration\_influenced))  
  
# Prepare variables for tabset (initialize as NULL)  
p <- NULL  
summary\_stats <- NULL  
  
# Step 11: Enhanced network visualisation and summary statistics   
if(length(total\_influenced\_collaborators) > 0) {  
   
 # Collect all relevant nodes for visualization  
 all\_pathway\_works <- unique(c(  
 sailor\_ivy\_works,  
 direct\_influence$from, direct\_influence$to,  
 two\_step\_pathways$sailor\_work, two\_step\_pathways$intermediate\_work, two\_step\_pathways$oceanus\_work,  
 cross\_collab\_influence$from, cross\_collab\_influence$to  
 ))  
   
 all\_relevant\_people <- unique(c(  
 sailor\_vertex\_name,  
 ivy\_echos\_name,  
 total\_influenced\_collaborators,  
 shared\_collaborators,  
 bridge\_to\_oceanus$bridge\_person  
 ))  
   
 all\_viz\_nodes <- unique(c(all\_pathway\_works, all\_relevant\_people))  
   
 # Enhanced node classification  
 viz\_nodes <- mc1\_nodes\_clean %>%  
 filter(name %in% all\_viz\_nodes) %>%  
 mutate(  
 influence\_strength = case\_when(  
 name %in% direct\_influence$to ~ "Direct Target",  
 name %in% two\_step\_pathways$oceanus\_work ~ "Indirect Target",   
 name %in% cross\_collab\_influence$to ~ "Secondary Target",  
 name %in% shared\_collaborators ~ "Shared Collaborator",  
 name %in% bridge\_to\_oceanus$bridge\_person ~ "Bridge Collaborator",  
 TRUE ~ "Network Node"  
 ),  
 node\_role = case\_when(  
 name == sailor\_vertex\_name ~ "Sailor Shift",  
 name == ivy\_echos\_name ~ "Ivy Echos",  
 name %in% sailor\_ivy\_works ~ "Sailor/Ivy Work",  
 name %in% oceanus\_folk\_works ~ "Oceanus Folk Work",  
 name %in% total\_influenced\_collaborators ~ "Influenced Oceanus Collaborator",  
 name %in% oceanus\_folk\_collaborators ~ "Other Oceanus Collaborator",  
 name %in% intermediate\_works ~ "Intermediate Work",  
 TRUE ~ "Other"  
 ),  
 node\_color = case\_when(  
 node\_role == "Sailor Shift" ~ "red",  
 node\_role == "Ivy Echos" ~ "purple",   
 node\_role == "Sailor/Ivy Work" ~ "gray30",  
 influence\_strength == "Direct Target" ~ "darkred",  
 influence\_strength == "Indirect Target" ~ "orange",  
 influence\_strength == "Secondary Target" ~ "yellow",  
 influence\_strength == "Shared Collaborator" ~ "blue",  
 influence\_strength == "Bridge Collaborator" ~ "cyan",  
 node\_role == "Influenced Oceanus Collaborator" ~ "darkgreen",  
 node\_role == "Other Oceanus Collaborator" ~ "lightgreen",  
 node\_role == "Intermediate Work" ~ "pink",  
 TRUE ~ "lightgray"  
 ),  
 node\_size = case\_when(  
 node\_role %in% c("Sailor Shift", "Ivy Echos") ~ 8,  
 influence\_strength %in% c("Direct Target", "Shared Collaborator") ~ 6,  
 influence\_strength %in% c("Indirect Target", "Bridge Collaborator") ~ 5,  
 influence\_strength == "Secondary Target" ~ 4,  
 TRUE ~ 3  
 ),  
 tooltip\_text = paste0(  
 "Name: ", node\_name, "\n",  
 "Role: ", node\_role, "\n",   
 "Influence: ", influence\_strength, "\n",  
 "Type: ", `Node Type`, "\n",  
 ifelse(!is.na(genre), paste0("Genre: ", genre), "")  
 )  
 )  
   
 # Collect all relevant edges preserving original Edge Types  
 all\_influence\_edges <- bind\_rows(  
 direct\_influence %>% mutate(pathway\_category = "Direct"),  
 two\_step\_pathways %>%   
 select(from = sailor\_work, to = intermediate\_work, `Edge Type` = step1\_edge\_type) %>%  
 mutate(pathway\_category = "Indirect Step 1"),  
 two\_step\_pathways %>%   
 select(from = intermediate\_work, to = oceanus\_work, `Edge Type` = step2\_edge\_type) %>%  
 mutate(pathway\_category = "Indirect Step 2"),  
 cross\_collab\_influence %>%   
 select(from, to, `Edge Type`) %>%  
 mutate(pathway\_category = "Cross-Collaborator")  
 )  
   
 viz\_edges <- mc1\_edges\_clean %>%  
 filter(from %in% all\_viz\_nodes, to %in% all\_viz\_nodes) %>%  
 left\_join(  
 all\_influence\_edges %>% select(from, to, pathway\_category),  
 by = c("from", "to")  
 ) %>%  
 mutate(  
 # Categorize edges for visual emphasis while keeping original Edge Type  
 edge\_category = case\_when(  
 !is.na(pathway\_category) ~ "Influence Pathway",  
 `Edge Type` == "MemberOf" & from == sailor\_vertex\_name ~ "Key Membership",   
 `Edge Type` %in% collab\_credit\_types ~ "Collaboration",  
 `Edge Type` %in% influence\_edge\_types ~ "Other Influence",  
 TRUE ~ "Other"  
 ),  
 edge\_alpha = case\_when(  
 edge\_category == "Influence Pathway" ~ 0.9,  
 edge\_category == "Key Membership" ~ 0.8,  
 edge\_category == "Collaboration" ~ 0.4,  
 edge\_category == "Other Influence" ~ 0.6,  
 TRUE ~ 0.2  
 )  
 )  
   
 # Create network plot  
 influence\_graph <- tbl\_graph(nodes = viz\_nodes, edges = viz\_edges, directed = TRUE)  
  
 layout\_df <- create\_layout(influence\_graph, layout = "fr") %>%  
 as\_tibble() %>%  
 select(name, x, y)  
   
 nodes\_plot <- as\_tibble(influence\_graph) %>%  
 left\_join(layout\_df, by = "name")  
   
 edges\_plot <- viz\_edges %>%  
 left\_join(nodes\_plot %>% select(name, x, y), by = c("from" = "name")) %>%  
 rename(x\_from = x, y\_from = y) %>%  
 left\_join(nodes\_plot %>% select(name, x, y), by = c("to" = "name")) %>%  
 rename(x\_to = x, y\_to = y)  
   
 # Create legend data frame for node colors  
 legend\_data <- data.frame(  
 node\_color = c("red", "purple", "gray30", "darkred", "orange", "yellow",   
 "blue", "cyan", "darkgreen", "lightgreen", "pink", "lightgray"),  
 node\_role = c("Sailor Shift", "Ivy Echos", "Sailor/Ivy Work", "Direct Target",   
 "Indirect Target", "Secondary Target", "Shared Collaborator",   
 "Bridge Collaborator", "Influenced Oceanus Collaborator",   
 "Other Oceanus Collaborator", "Intermediate Work", "Other"),  
 stringsAsFactors = FALSE  
 )  
   
 p <- ggplot() +  
 geom\_segment(  
 data = edges\_plot,  
 aes(x = x\_from, y = y\_from, xend = x\_to, yend = y\_to,  
 color = `Edge Type`, alpha = edge\_alpha),  
 arrow = arrow(length = unit(1.5, 'mm'))  
 ) +  
 scale\_alpha\_identity() +  
 scale\_color\_discrete(name = "Edge Type") +  
 ggnewscale::new\_scale\_color() +  
 geom\_point\_interactive(  
 data = nodes\_plot,  
 aes(x = x, y = y, tooltip = tooltip\_text, data\_id = name,  
 color = node\_color, shape = `Node Type`, size = node\_size)  
 ) +  
 scale\_size\_identity() +   
 scale\_color\_manual(  
 name = "Node Role",  
 values = c("red" = "red", "purple" = "purple", "pink" = "pink", "darkred" = "darkred",  
 "orange" = "orange", "yellow" = "yellow", "blue" = "blue", "cyan" = "cyan",   
 "darkgreen" = "darkgreen", "lightgreen" = "lightgreen",  
 "gray30" = "gray30", "lightgray" = "lightgray"),  
 labels = setNames(legend\_data$node\_role, legend\_data$node\_color),  
 breaks = legend\_data$node\_color,  
 guide = guide\_legend(override.aes = list(size = 4, shape = 16))  
 ) +  
 geom\_text(  
 data = nodes\_plot %>% filter(node\_role == "Sailor Shift"),  
 aes(x = x, y = y, label = "Sailor Shift"),  
 size = 4, fontface = "bold", color = "red", vjust = -2  
 ) +  
 theme\_void() +  
 theme(  
 legend.position = "right",  
 legend.box = "vertical",  
 legend.text = element\_text(size = 11),  
 legend.title = element\_text(size = 16),  
 plot.title = element\_text(size = 20, face = "bold"),   
 plot.subtitle = element\_text(size = 16, face = "plain")   
 ) +  
 labs(  
 title = "Sailor Shift's Influence on Oceanus Folk Community",  
 subtitle = str\_to\_title("Influence pathways: Direct (work-to-work), Indirect (via intermediary), Secondary (cross-collaborator), Shared/Bridge (collaboration networks)")  
 )  
   
 # Create summary statistics data frame  
summary\_stats <- data.frame(  
 Metric = c(  
 "Total Oceanus Folk collaborators",  
 "Total influenced collaborators",  
 "Percentage influenced (%)",  
 "",  
 "Direct influences",  
 "Two-step pathways",   
 "Cross-collaborator influences",  
 "Shared collaborators",  
 "Bridge collaborators"  
 ),  
 Value = c(  
 length(oceanus\_folk\_collaborators),  
 length(total\_influenced\_collaborators),  
 round(100 \* length(total\_influenced\_collaborators) / length(oceanus\_folk\_collaborators), 1),  
 "",  
 nrow(direct\_influence),  
 nrow(two\_step\_pathways),  
 nrow(cross\_collab\_influence),  
 length(shared\_collaborators),  
 length(unique(bridge\_to\_oceanus$bridge\_person))  
 ),  
 stringsAsFactors = FALSE  
)  
}

## Network Visualisation

girafe(ggobj = p, width\_svg = 16, height\_svg = 12)



## Summary Statistics

knitr::kable(  
 summary\_stats,  
 col.names = c("Metric", "Count"),  
 caption = "Sailor Shift's Influence Analysis Summary"  
)

Sailor Shift’s Influence Analysis Summary

| Metric | Count |
| --- | --- |
| Total Oceanus Folk collaborators | 720 |
| Total influenced collaborators | 81 |
| Percentage influenced (%) | 11.2 |
|  |  |
| Direct influences | 9 |
| Two-step pathways | 11 |
| Cross-collaborator influences | 10 |
| Shared collaborators | 42 |
| Bridge collaborators | 7 |

The above visualisation focus on the network of influence that Sailor Shift had in the Oceanus Folk community. Out of 720 Oceanus Folk collaborators, she has interacted with 81 collaborators, which is a notable influence as it is more than 10% of the community.

While only 9 collaborators have been directly influenced by working closely with her/Ivy Echos, the majority of her impact is indirect. More than half of the influenced collaborators have been shaped indirectly through shared and bridged collaborators. These network-mediated pathways, including two-step and cross-collaborator connections, illustrates how her influence extends beyond those that she worked directly with.

Overall, this showcases how Sailor Shift’s influence diffuses dynamically throughout the community where her impact in the community is not only driven by direct collaborations, but also by the broader web of relationships and interactions that connect the Oceanus Folk community.

# 6 References

* [VAST Challenge 2025](https://vast-challenge.github.io/2025/MC1.html)
* [PageRank](https://cambridge-intelligence.com/keylines-faqs-social-network-analysis/)
* [PageRank - Calculate node and edge centrality](https://tidygraph.data-imaginist.com/reference/centrality.html)