

## RAPPORT DE INFORMATION VISUALISATION

---

# Visualisations d'une base de données musicale

---

*Réalisé par*  
**Mia Swery**  
**Eva Radu**  
**Hugo Bulzomi**

*Encadré par*  
**Marco Winckler**  
**Aline Menin**

**Table des matières**

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Présentation du sujet . . . . .	2
1.2	Outils utilisés . . . . .	2
<b>2</b>	<b>Développement du projet</b>	<b>2</b>
2.1	Pré-traitement des données . . . . .	2
2.2	Choix des visualisations . . . . .	3
2.3	Visualisation via un graphe . . . . .	3
2.3.1	But de la visualisation . . . . .	3
2.3.2	Pré-traitement . . . . .	3
2.3.3	Que permet la visualisation ? . . . . .	4
2.4	Visualisation via un sunburst . . . . .	5
2.4.1	But de la visualisation . . . . .	5
2.4.2	Pré-traitement . . . . .	5
2.4.3	Que permet la visualisation ? . . . . .	6
2.5	Visualisation via des coordonnées parallèles . . . . .	10
2.5.1	But de la visualisation . . . . .	10
2.5.2	Pré-traitement . . . . .	10
2.5.3	Que permet la visualisation ? . . . . .	12
2.5.4	Fonctionnement en interne . . . . .	14
<b>3</b>	<b>Conclusion</b>	<b>15</b>

## 1 Introduction

### 1.1 Présentation du sujet

Le but de ce projet est de permettre la représentation d'informations contenues dans une base de données WASABI en utilisant différentes techniques de visualisation.

La base de données en question, créée par Michel Buffa, contient des informations relatives à 3 grandes entités : des artistes, des albums et des chansons. On a à disposition un certain nombre de détails propres à chaque entité. Par exemple, le nombre de followers d'un artiste sur Deezer, la langue d'une chanson, la date de publication d'un album...

Étant donnée la quantité importante d'information présente dans la base de données, il a fallu dans un premier temps décider sous quel angle nous allions aborder nos visualisations. Il nous a ainsi paru judicieux de se concentrer sur la notion de "genre". En effet, toutes nos entités ont des genres associés et nous pourrions donc tenter de représenter ce phénomène via nos visualisations.

Pour mener à bien cette tâche, nous avons formé notre équipe de deux étudiantes en IHM et d'un étudiant en IA. En combinant l'ensemble de nos compétences, nous espérons donc réaliser au mieux notre projet.

### 1.2 Outils utilisés

En ce qui concerne les outils utilisés, nous avons décidé d'écrire nos scripts de traitement des données en langage Python ou R. De plus, pour la partie de l'implémentation des visualisations, nous avons utilisé d3.js, HTML et CSS. Nous n'avons pas utilisé R Shiny, bien que nous l'aurions voulu. En effet, nous avions énormément de mal à trouver des exemples pour nous aider dans la conception de nos visualisations.

## 2 Développement du projet

L'ensemble du code est disponible via ce lien GitHub : <https://github.com/EvaRadu/Music-Visualization.git>

De plus, il est à noter que les fichiers de données étaient trop volumineux pour être placés sur le répertoire GitHub. Nous avons donc conservé les jeux de données localement sur nos ordinateurs respectifs dans un dossier nommé "DATA". Toutefois, via ce lien, il est possible de télécharger nos fichiers de données : <https://we.tl/t-OAGFRunwgG>

### 2.1 Pré-traitement des données

Tout d'abord, nous avons dû obtenir des jeux de données exploitables. Nous avons donc transformé les fichiers au format rds dont nous disposions au format csv. C'est grâce à la fonction présente dans le fichier rdsToCsv.R que nous avons pu réaliser cette tâche.

Il a fallu par la suite exploiter nos données.

On s'est aperçu d'un problème : l'absence de données. En effet, certains albums ou chansons n'ont aucun genre, ce qui risque de poser un problème étant donné que nous voulions nous concentrer sur les genres pour nos visualisations. Pour ce faire, nous avons fait le choix d'inférer le genre d'un album en fonction de son artiste et celui d'une chanson en fonction de son album. De cette manière, on arrive à combler certains vides. En revanche, on ne peut inférer un genre à un artiste et de fait, on ne peut inférer un genre à l'un de ses albums ou chansons. Nous avons effectué tout cela grâce au script "readCsv.py".

## 2.2 Choix des visualisations

Nous avons décidé d'implémenter les trois visualisations suivantes :

- Un Graphe (implémenté par Hugo)
- Un Sunburst (implémenté par Mia)
- Des Coordonées Parallèles (implémenté par Eva)

Nous avons alors chacun créé une page HTML pour notre visualisation.

De plus, nous avons créé une page HTML nommée "musicVisualisation.html" qui permet de choisir entre les trois visualisations grâce au menu horizontal suivant :



Une fois la visualisation choisie, nous sommes redirigé vers la page HTML correspondante. Il est ensuite possible de basculer d'une page HTML à une autre grâce à ce même menu.

## 2.3 Visualisation via un graphe

### 2.3.1 But de la visualisation

Dans le contexte de notre problématique, nous pouvons dégager de notre dataset des relations entre des entités : les artistes/albums/chansons étant reliés à des genres/sous-genres. Il est alors naturel d'envisager un graphe comme technique de visualisation. Un graphe simple ne permet néanmoins pas de visualiser efficacement de grands ensembles de données, le tout devenant très vite illisible pour un utilisateur. Nous avons ainsi décidé d'utiliser un arbre dépliable. Le but est de permettre à l'utilisateur de progressivement naviguer depuis des informations globales et peu détaillées, vers des informations plus précises en n'affichant que les parties du graphe nécessaires. L'utilisateur commencera donc par sélectionner un genre, puis un sous-genre, un artiste, et enfin un album afin de progressivement afficher les informations pertinentes sans surcharger l'écran. La figure 1 montre les noeuds et relations utilisés dans l'arbre.

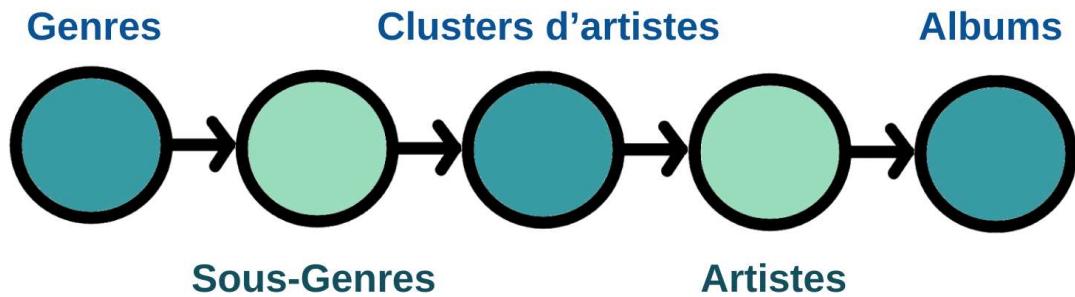


FIGURE 1 - Relations utilisées dans l'arbre dépliant.

Les noeuds "clusters d'artistes" sont des clusters de 50 artistes groupés alphabétiquement afin de réduire la largeur de l'arbre pour le rendre plus lisible. On remarque que les chansons ne sont pas des noeuds : on peut trouver ces dernières dans la vue détaillée des albums lorsque l'utilisateur clique dessus

### 2.3.2 Pré-traitement

Le pré-traitement spécifique à ce graphe a consisté à recréer les relations de parentalité dans un fichier JSON.

- Les genres ont pour enfants leurs sous-genres associés.
- Les sous-genres sont reliés à des clusters d'artistes faisant partie de ce sous-genre (groupes de 50 artistes par ordre alphabétique pour rendre le tout plus lisible).
- Les clusters d'artistes sont reliés aux artistes individuels qui les constituent.
- Enfin, chaque artiste a pour enfants ses albums.

Dans un second document Json, chaque album a aussi été associé aux chansons qui le constituent afin de pouvoir afficher ces dernières sur demande de l'utilisateur dans un onglet de visualisation détaillée.

À noter que cette visualisation utilise les genres inférés : une chanson sans genre prendra celui/-ceux de son album, et un album prendra celui/ceux de son artiste.

### 2.3.3 Que permet la visualisation ?

Comme détaillé plus haut, la visualisation permet à l'utilisateur de cliquer sur un noeud pour en afficher les enfants, et ainsi d'afficher des informations de plus en plus spécifiques. Au moment où un sous-genre est sélectionné, le noeud de ce dernier s'affiche en vert. On peut alors utiliser la barre de recherche afin de rapidement déplier la partie pertinente de l'arbre pour trouver l'artiste recherché dans le sous-genre sélectionné. Les noeuds automatiquement dépliés par la recherche se colorent en rouge pour montrer l'arborescence qui aboutit à l'artiste recherché. Les noeuds des albums faisant partie du sous-genre sélectionné s'affichent eux aussi en vert. La figure 2 montre un exemple de dépliage d'arbre après recherche d'un artiste dans un sous-genre.

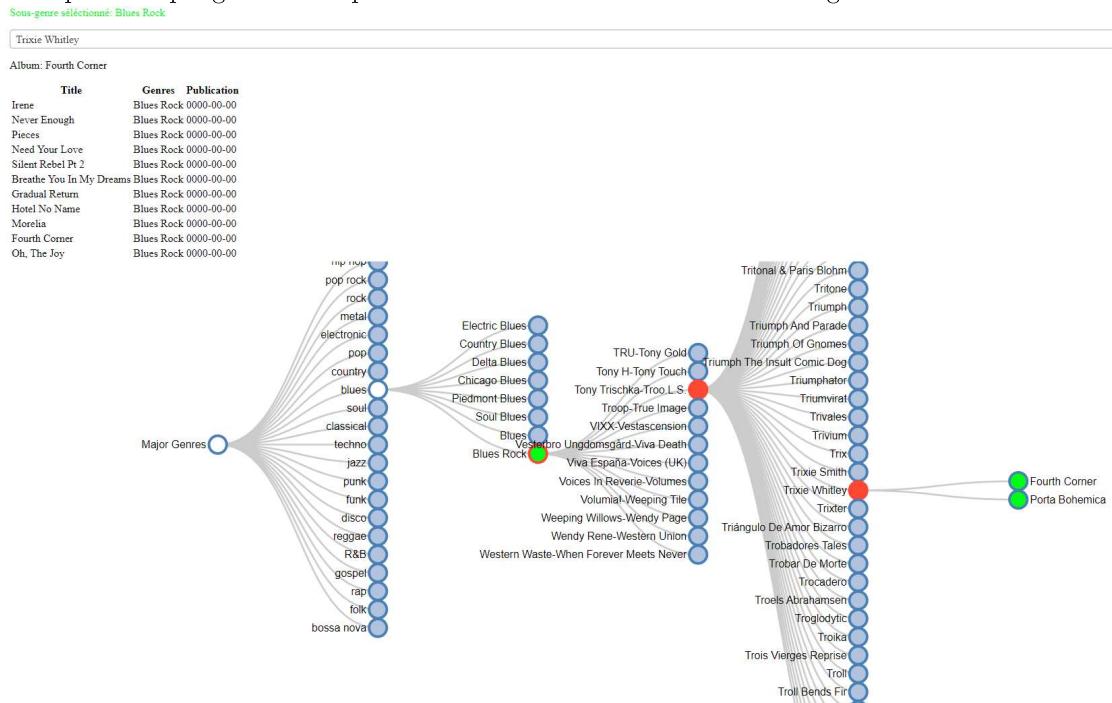


FIGURE 2 - Exemple de dépliage de l'arbre.

On peut rapidement voir que les deux albums de l'artiste Trixie Whitley font partie du sous-genre Blues Rock sélectionné. En haut à gauche les détails de l'album sur lequel l'utilisateur a cliqué sont visibles, avec les chansons qui le constituent, et différentes informations.

La recherche d'un artiste se fait toujours dans le sous-genre sélectionné par l'utilisateur. Si aucun artiste avec le nom spécifié n'est trouvé dans le sous-genre, un message d'erreur s'affiche pour informer l'utilisateur.

La présentation de la visualisation via un graphe étant terminée, nous allons maintenant nous attarder plus en détails sur la visualisation utilisant un sunburst.

## 2.4 Visualisation via un sunburst

### 2.4.1 But de la visualisation

L'objectif de cette visualisation est de permettre aux utilisateurs de déterminer le genre prépondérant d'albums produits dans une décennie donnée. Pour ce faire, nous allons utiliser une technique de visualisation hiérarchique : un sunburst.

### 2.4.2 Pré-traitement

Étant donné qu'un sunburst est une méthode de visualisation supposant une relation hiérarchique entre nos éléments, il a fallu dans un premier temps construire cette hiérarchie. La construction d'un json similaire à celui utilisé pour la visualisation avec un graphe nous semblait donc être judicieux. Toutefois, assez rapidement, nous avons rencontré un problème difficile à résoudre : l'absence totale de tutorial permettant de construire un sunburst à partir d'un fichier json en d3.js. Il a donc finalement dû se résoudre à créer un fichier csv, plutôt qu'un json. On a donc dû réadapter l'ensemble de la phase de pré-traitement réalisée en commun à la création d'un csv appelé "sunburst.csv" contenant les informations recherchées. Dans un premier temps, il a fallu faire certains choix concernant ce qui allait être représenté. Fallait-il partir sur une année en particulier ou bien une période précise ? Nous avons choisi de travailler par tranche de dix ans de 1970 à 2010. Il a donc fallu d'abord associer une décennie à la date de publication de chaque album : en ajoutant une colonne spécifique dans le data frame contenant l'ensemble des informations que nous avons jugées utile liés aux albums, à savoir : id, name, genre, id\_artist, title, publicationDate, deezerFans, dateRelease, language, genre\_infere.

Par la suite, on a repris les méthodes employées dans la phase de pré-traitement commune pour faire des liens entre genre, sous-genre et genre\_inféré. Dans le cas, où un album n'a pas de genre défini, on associe à un album le genre de son artiste. Le soucis étant que bien souvent, il s'agit plus d'une liste de genre que d'un genre qui est inféré. C'est pour cela qu'il a paru judicieux de créer un dataframe contenant un genre par album par ligne (quitte à ce qu'un album soit présent sur plusieurs lignes).

Il a fallu aussi retrouver le nom d'un artiste ayant réalisé un album. En effet, dans le dataframe contenant les informations relatives à un album, on connaît l'id d'un artiste mais pas son nom. Ainsi, à partir de l'id de l'artiste associé à l'album, on peut retrouver son nom dans notre data frame avec les informations liées aux artistes. On pourra ainsi ajouter cette information à notre sunburst. On insère les noms des artistes dans le dataframe des albums.

Notre dataframe Albums contenant désormais l'ensemble des informations dont on a besoin pour construire notre visualisation, il est donc temps de passer à la phase de création de notre fichier csv.

À l'aide de la méthode creationCsv() dans le fichier preprocessing.py, on va définir notre fichier csv. Pour ce faire, on va construire une liste de listes, où chaque liste correspond à une ligne (row) de notre fichier csv. Elle se construit comme suit : [id, name, idParent, value].

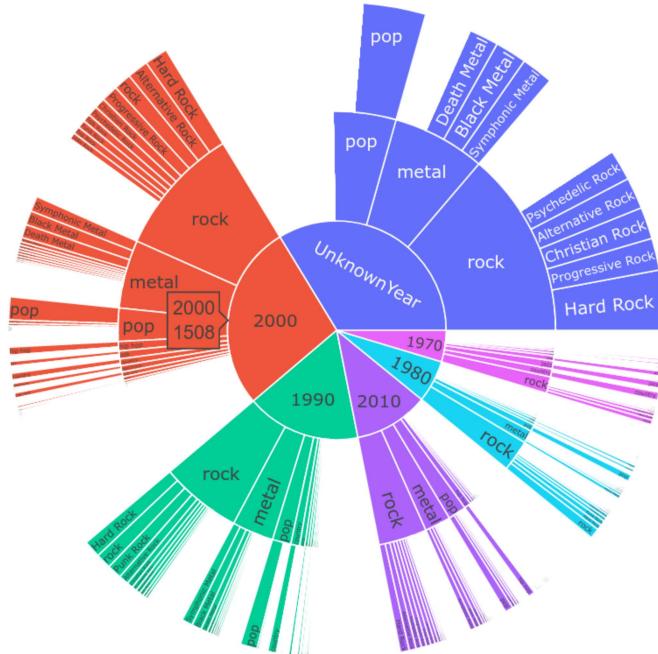
- Les ids sont nécessaires pour qu'un élément soit unique et donc qu'il n'y ait pas d'ambiguïté lorsqu'on va construire nos relations parents enfants.

- Name est le contenu du noeud.
- IdParent est l'id du parent de l'élément actuel.
- Enfin value correspond à la proportion de l'élément actuel dans le sunburst.

On va donc parcourir la liste de genres et des sous genres associés par décennie. Pour chaque sous-genre, on cherche le nombre d'album de genre ou de sous-genre inféré identique. On cherche ensuite à lier les sous-genre au genre puis à l'année via des boucles itératives et les ids. Une fois cette phase de construction terminée, nous obtenons un fichier csv. Ce dernier est utilisé par notre script JavaScript qui va insérer un sunburst dans notre page html dédiée à la visualisation du sunburst (en appuyant sur un bouton "submit" qui lance la création en interne de cette visualisation).

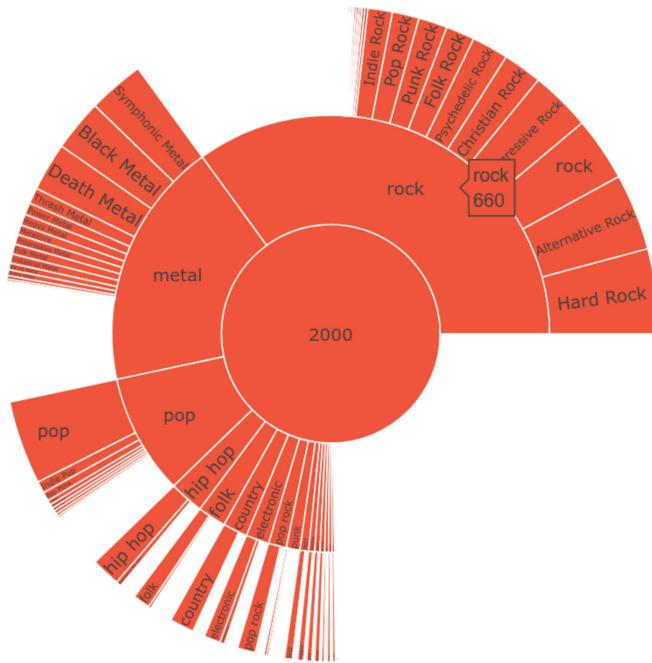
#### 2.4.3 Que permet la visualisation ?

Au terme de l'étape de pré-traitement des données, nous obtenons finalement bien un sunburst. Il se présente ainsi :



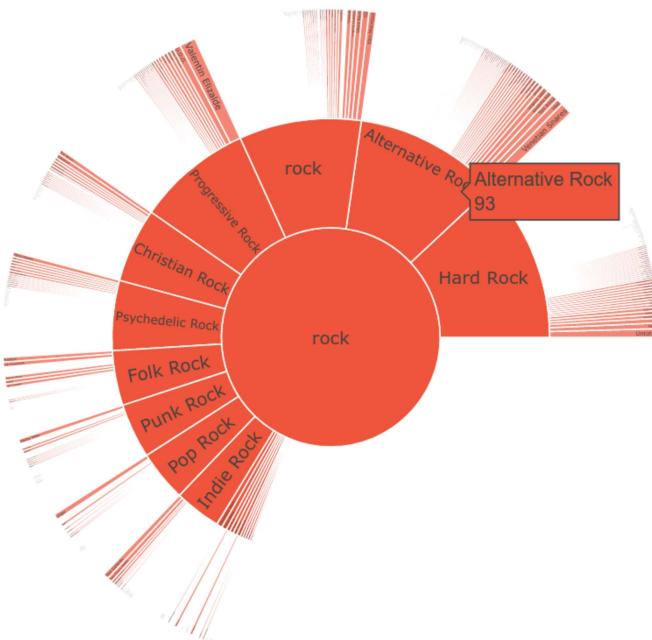
Des tooltips popups apparaissent à l'écran. Ils indiquent le nombre d'albums produits dans la catégorie. Par exemple, sur la capture d'écran ci-dessus, on s'aperçoit que dans les années 2000, 1508 albums ont été produits.

Il est possible de "zoomer" les informations. On peut en effet cliquer sur une décennie et obtenir en détail les albums produit par genre et sous genre. Si on zoom une fois de plus, on peut même regarder le nom des artistes ayant produit des albums d'un sous genre en particulier et le nombre exact d'albums produits par artiste. Simulons un utilisateur souhaitant s'attarder sur la visualisation d'information sur le genre "rock" dans la décennie 2000. On peut, dans un premier temps, appuyer sur la tranche année 2000. Le résultat est :



Grâce à la tooltip popup qui apparaît quand on passe le curseur sur la zone souhaitée (ici "rock"), on sait que parmi les 1508 albums produits dans la décennie 2000, 660 ont pour genre "rock".

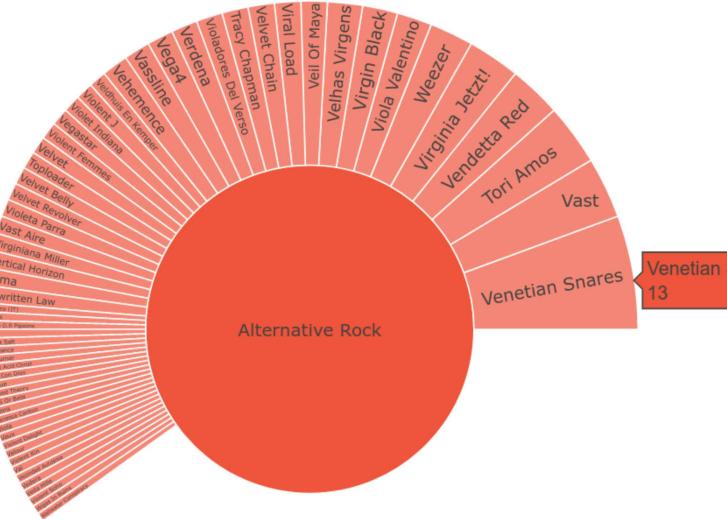
Mettons qu'on veuille davantage de détails sur le genre "rock" dans les années 2000, il suffit de cliquer dessus. On obtient alors :



On s'aperçoit donc que l'un des sous-genres majoritaires est "Alternative rock" avec 93 albums

produits.

En appuyant sur cette catégorie, on obtient :



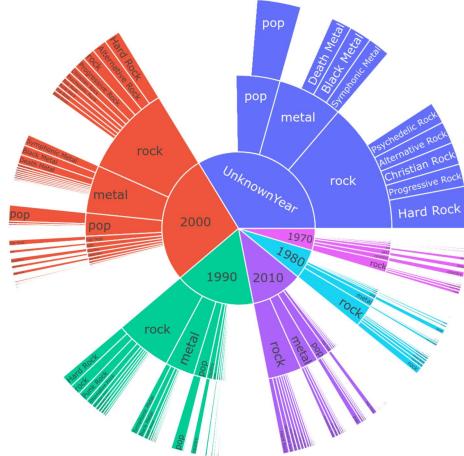
Il est bien plus facile de lire les noms des artistes ayant produit des albums de cette catégorie. Ici, on s'aperçoit que c'est le groupe "Venetian snares" qui a produit le plus d'album du sous-genre "Alternative Rock" dans la décennie 2000 avec 13 albums produits.

En plus du 'zoom', on a aussi la possibilité d'appliquer des filtres à notre visualisation, grâce au menu présent en haut de page. Lorsque tous les filtres sont activés (ce qui est le cas par défaut), nous obtenons :



### Sunburst

- Afficher les données liées aux années 1970
- Afficher les données liées aux années 1980
- Afficher les données liées aux années 1990
- Afficher les données liées aux années 2000
- Afficher les données liées aux années 2010

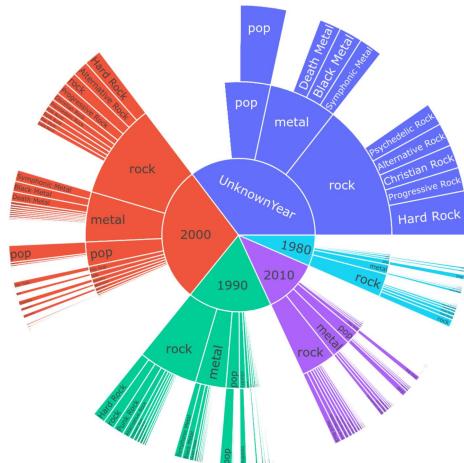


Toutefois, je peux très bien faire le choix de ne pas afficher une décennie en particulier, la décennie 1970 par exemple :

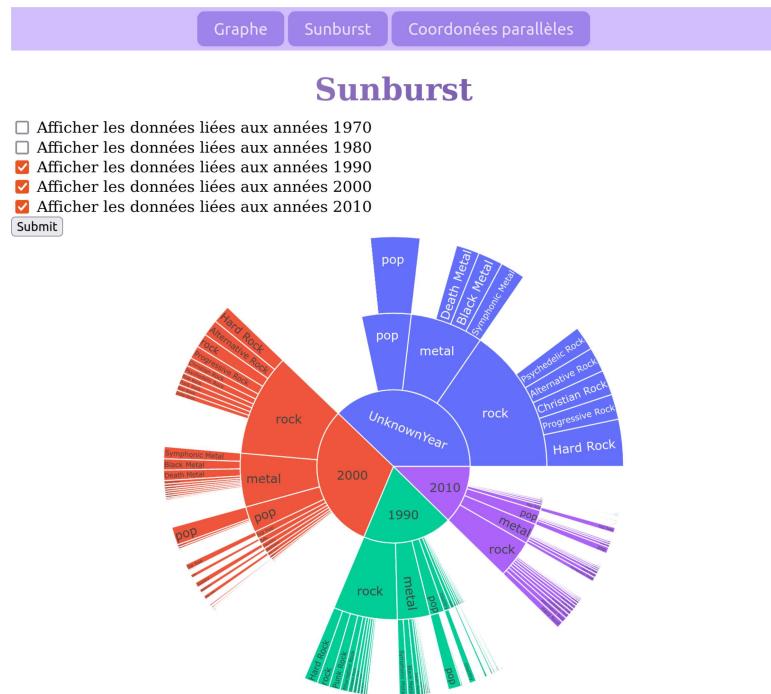


### Sunburst

- Afficher les données liées aux années 1970
- Afficher les données liées aux années 1980
- Afficher les données liées aux années 1990
- Afficher les données liées aux années 2000
- Afficher les données liées aux années 2010



Il est également possible de désélectionner plusieurs filtres en même temps. Par exemple, je ne souhaite pas visualiser les informations relatives aux décennies 1970 et 1980. On obtient :



Nous arrivons au terme de nos explications sur la visualisation via un sunburst. Nous allons désormais présenter notre visualisation utilisant des coordonnées parallèles.

## 2.5 Visualisation via des coordonnées parallèles

### 2.5.1 But de la visualisation

Cette visualisation a pour but de permettre aux utilisateurs d'accéder aux détails des musiques réalisées par un artiste qui aura été sélectionné. La visualisation permet notamment de voir les genres ou genres inférés les plus récurrents parmi les chansons d'un artiste.

La sélection d'un artiste était nécessaire pour cette visualisation au vu du nombre très important de chansons disponibles dans le dataset original. En effet, plus de 76000 chansons y sont référencées, leur affichage aurait alors été complètement illisible dans une même visualisation de coordonnées parallèles.

### 2.5.2 Pré-traitement

Plusieurs actions ont dû être effectuées en amont afin de pouvoir réaliser la visualisation. Tout cela a été réalisé grâce au script python “preProcessing.py” du dossier “Parallel Coordinates”. Des modifications sur les fichiers songs.csv ont été effectuées et sauvegardées dans le fichier songs\_parallelCoord.csv.

Premièrement, toutes les données manquantes ont été remplacées par “Unknown”. Cela était nécessaire pour pouvoir permettre l'accès à ces informations dans la visualisation. En effet, il est important d'afficher toutes les données du dataset, même celles manquantes. De plus, un tri sur les colonnes a été effectué. En effet, la grande majorité des colonnes a été supprimée étant donné qu'elles n'apportaient pas d'intérêt dans la visualisation car presque toutes leurs valeurs étaient manquantes.

Par la suite, le nom de l'artiste et le nom de l'album ont été rajoutés pour chaque chanson. L'ajout de ces deux informations a été fait ainsi : tout d'abord on récupère l'identifiant de l'album de la musique dans le fichier songs.csv, puis on regarde le nom de l'album auquel il correspond dans albums.csv et enfin on récupère l'identifiant de l'artiste ayant fait cet album et on regarde le nom qui lui correspond dans artists.csv.

De plus, avec le jeu de données des chansons original, il n'était pas possible de réaliser la visualisation étant donné que chaque chanson possédait une liste de genres ou une liste de genres inférés. Cependant, pour réaliser une représentation en coordonnées parallèles, il est nécessaire de n'avoir qu'un élément par colonne dans le fichier csv. Une première étape de prétraitement a donc consisté à dupliquer les lignes du fichier songs.csv afin de n'avoir qu'un genre ou qu'un genre inféré par ligne. Par exemple, les deux lignes suivantes :

id	title	langue	genre	genre inféré	...
5714deea	Alice Blue Gown	Unknown	Unknown	Trip Hop, Soul, Rock	...
5737ed	Please Love Me Forever	Unknown	Trip Hop, Soul	Unknown	...

Ont été dupliqué de la façon suivante :

id	title	langue	genre	genre inféré	...
5714deea	Alice Blue Gown	Unknown	Unknown	Trip Hop	...
5714deea	Alice Blue Gown	Unknown	Unknown	Soul	...
5714deea	Alice Blue Gown	Unknown	Unknown	Rock	...
5737ed	Please Love Me Forever	Unknown	Trip Hop	Unknown	...
5737ed	Please Love Me Forever	Unknown	Soul	Unknown	...

Cependant, après avoir réalisé cette duplication des lignes, la taille du fichier était beaucoup trop grande : cela posait donc des problèmes d'interactivité. En effet, une fois l'interface mise en place, lorsqu'un utilisateur sélectionnait un artiste, l'affichage de la visualisation correspondante prenait beaucoup de temps étant donné que le programme cherchait parmi toutes les lignes de ce fichier les lignes correspondant à l'artiste en question.

Afin de pallier ce problème, j'ai séparé ce fichier en plein de petit fichier. En effet, un fichier par artiste a été créé : chaque fichier regroupe donc toutes les chansons d'un artiste en particulier. Nous avons donc obtenu 3000 fichiers, qui ont été placés dans le dossier "ParCoordCsv" du dossier DATA. Ainsi, une fois un artiste sélectionné, il nous suffit en interne de chercher le fichier correspondant à cet artiste afin d'obtenir les informations des musiques correspondantes rapidement. Nous sommes conscients que cette méthode coûte en mémoire, mais il était nécessaire de faire un choix entre beaucoup de données en mémoire ou un manque d'interactivité au sein

de la visualisation.

### 2.5.3 Que permet la visualisation ?

Comme évoqué précédemment, cette visualisation permet l'accès aux détails des musiques réalisées par un artiste.

La première étape consiste donc à choisir un artiste parmi la liste des artistes. Voici le menu permettant cette action :

## Coordonées parallèles

Veuillez selectionner un artiste :

Une fois l'artiste sélectionné, plusieurs filtres peuvent être appliqués. Deux types de filtres ont été implémentés :

- Des filtres permettant de retirer des colonnes dans la visualisation en coordonnées parallèles
- Des filtres permettant de ne pas afficher les données manquantes (c'est-à-dire les données "Unknown")

Voici le menu permettant de sélectionner les filtres souhaités :

Selection de filtres :

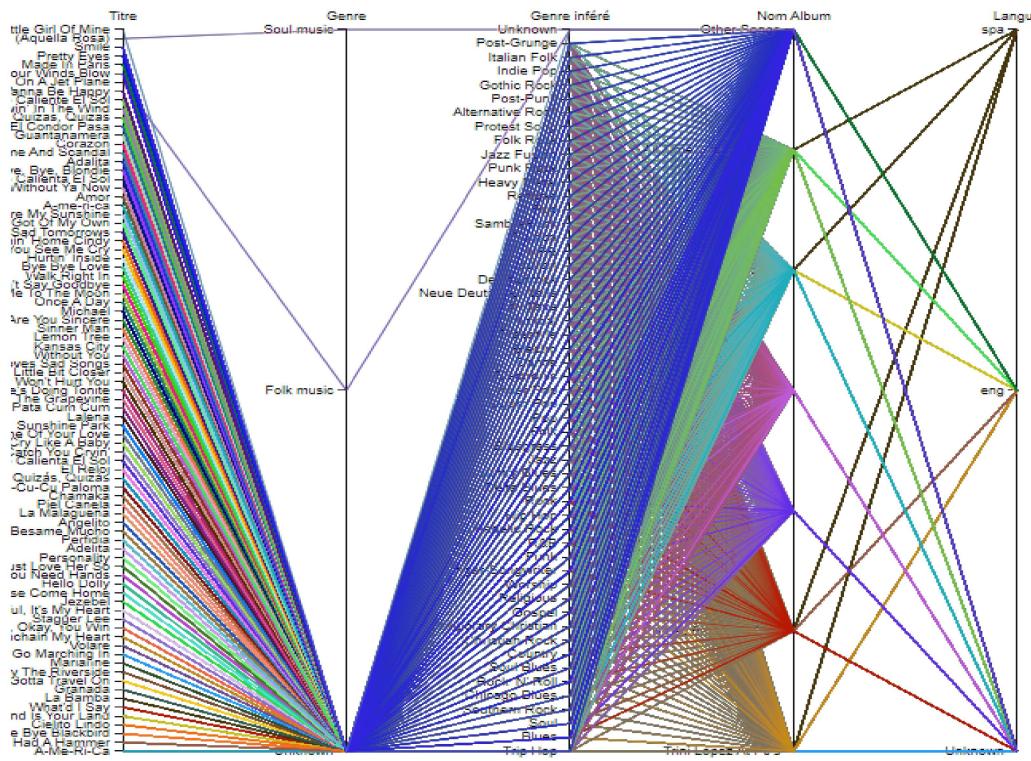
- Afficher la colonne des genres
- Afficher la colonne des genres inférés
- Afficher la colonne des albums
- Afficher la colonne des langues
- Afficher les données avec un genre inconnu
- Afficher les données avec un genre inféré inconnu
- Afficher les données avec une langue inconnue

Confirmer

À noter que par défaut, toutes les colonnes sont affichées et toutes les données, même celles manquantes, sont affichées.

Après avoir confirmé sa sélection, l'utilisateur dispose alors de la visualisation en coordonnées parallèles correspondante.

Voici un exemple de visualisation qu'il est possible d'obtenir :



Dans l'exemple ci-dessus, l'artiste choisi (Trini Lopez) a beaucoup de chansons associées. Afin de rendre les informations plus visibles, l'utilisateur dispose de 3 solutions :

- En passant avec la souris sur le nom d'un élément (par exemple le nom d'une musique, d'un album, d'un genre etc ou bien sur le nom d'une colonne), il est possible d'agrandir le texte associé.
- Il est possible de zoomer et de naviguer dans la visualisation afin d'accéder plus en détail à n'importe quelle information ou n'importe quelle ligne.
- Enfin, en passant la souris sur un des traits, il est possible d'obtenir les informations associées. Ces informations sont affichées à droite de la visualisation dans une section prévue à cet effet. Voici la manière dont les informations sont affichées sur la droite de la visualisation :

**Details de la chanson :**

**Titre :** Leaving On A Jet Plane  
**Genre :** Folk music  
**Genre inféré :** Unknown  
**Nom Album :** Other Songs  
**Langue :** eng

De plus, en dessous de la visualisation, il est possible d'avoir une vue d'ensemble sur toutes les chansons de l'artiste sélectionné. Voici la manière dont les éléments sont affichés :



Enfin, il est à noter que si la sélection de filtres est associée à un ensemble de données vide, l'utilisateur est notifié de la sorte :



#### 2.5.4 Fonctionnement en interne

Dans le dossier Parallel Coordinates, en plus du fichier pour le traitement des données, on trouve 3 fichiers pour la visualisation :

- Le fichier parallelCoord.html
- Le fichier styleParallelCoord.css
- Le fichier parallelCoord.js

Au chargement de la page HTML, la fonction Javascript "readCsvArtist" est appelée avec en paramètre le fichier de données "artists.csv". Cette fonction a pour but de charger tous les noms des artistes dans le menu déroulant.

Par la suite, le bouton "Confirmer" de la page HTML est relié à la fonction Javascript "readCsvSong()" qui se charge de récupérer toutes les informations nécessaires pour la visualisation. Cette fonction prend en compte le nom de l'artiste sélectionné et les filtres choisis. Une fois toutes les informations nécessaires récupérées, cette fonction appelle la fonction "loadVisualization" qui supprime l'ancienne visualisation et charge la nouvelle visualisation à sa place. À noter que c'est en grande partie la fonction "loadVisualization" qui utilise d3.js.

### 3 Conclusion

Malgré les difficultés rencontrées, notamment due à notre faible expérience avec R Shiny et D3.js, on a tout de même réussi à obtenir trois visualisations fonctionnelles : un graphe, un sunburst et des coordonnées parallèles. Pour chaque visualisation proposée, il y a la possibilité d'appliquer des filtres.

De plus, la démarche effectuée a été très intéressante. En effet, à partir d'un même jeu de données, il a été possible, en utilisant trois techniques de visualisation différentes, de mettre en relief des informations variées.

La subtilité de ce projet réside finalement autant sur ce qu'on veut montrer que comment on veut le représenter.

Enfin, manipuler une librairie complexe telle que D3.js a été formateur, et nous sera sans doute utile plus tard dans notre vie professionnelle.