# Information Visualisation

Groupe BERNIGAUD-BOURDEAU-CORREIA-JEANNES :
Benjamin Bernigaud, Quentin Bourdeau
Ambre Correia, Theo Jeannes

## Introduction

Pour ce projet de visualisation, nous allons nous concentrer sur le caractère explicite ou non des chansons et l'évolution de la représentation de ces chansons dans l'industrie musicale. Tout au long du projet, nous étudierons donc la place des chansons à paroles explicites dans le monde de la musique.

Dans un premier temps, nous présenterons l'évolution de la proportion de chansons explicites produites à travers les années et les genres. Pour cela, nous utiliserons un graphique spirale ce qui nous permettra de mettre en évidence une éventuelle tendance dans l'évolution du nombre de chansons avec des paroles explicites au fil du temps.

Nous chercherons également à identifier la distribution des chansons explicites à travers différentes catégories. Nous représenterons donc sur des ensembles parallèles les chansons regroupées par genre, par albums, par artistes, genre de l'artiste ou du groupe, ainsi que par caractère explicite. Nous utiliserons le nombre de chansons de chaque catégorie pour définir l'importance des catégories dans le diagramme.

Ensuite, nous afficherons les chansons dans une pyramide de population pour déterminer si le caractère explicite d'une chanson influence sa popularité à travers les années. Pour cela, nous représenterons l'année de publication en ordonnée et la popularité moyenne d'une chanson explicite ou non en abscisse. Un zoom sémantique permettra d'afficher la répartition des genres ou des artistes dans les années affichées. Cela nous permettra de comparer la popularité des chansons explicites avec les chansons tout publics, et de détecter un éventuel changement dans l'appréciation des chansons explicites à travers le temps.

Enfin, notre dernière représentation prendra la forme d'une carte à bulles, ce qui nous permettra d'identifier les pays produisant le plus de chansons explicites. Cela nous permettra également de constater d'éventuelles disparités entre les zones géographiques, vis-a-vis de leur proportion de chansons explicites produites.

Le code source du traitement de données et des visualisations se situe dans l'annexe 1. Cette annexe inclut également un lien vers tous les fichiers .csv utilisés pendant le projet.

## Type de données visualisées

Pour nos représentations, nous allons devoir créer deux structures à partir du jeu de données:

- Pour pouvoir faire une carte à bulles, nous allons avoir besoin d'associer des positions à chaque point de données. Cela nous amènera à créer un champ de données.
- Pour représenter le diagramme en spirale, la pyramide de population et les ensembles parallèles, il nous suffira d'utiliser les données sous forme de table.

## Utilisateurs visés

Dans un premier temps, les labels, producteurs et artistes pourraient être intéressés par l'impact d'un contenu explicite sur sa popularité. Ils pourraient également chercher à savoir si une chanson explicite a autant de chances de devenir célèbre qu'une chanson sans paroles explicites.

Les législateurs et les autorités de régulation pourraient également être intéressés à comprendre dans quelle mesure les avertissements concernant les chansons explicites sont utiles. Ils pourraient également comparer l'évolution de la popularité des chansons explicites avec les réformes et lois sur la régulation de ces contenus pour évaluer l'efficacité de telles mesures.

## Récupération des données depuis WASABI

La première étape de ce projet a consisté à récupérer les données. Les données fournies ne contenaient pas le champ *explicitLyrics* pour les chansons, ce qui est crucial à nos visualisations. Pour cette raison, nous avons dû récupérer les données nous-même. Nous avons donc récupéré les données à travers l'API fournie. Pour cela, nous avons écrit un script Python, *importDatas.py*, permettant de faire des requêtes pour récupérer toutes les données nécessaires en adaptant dynamiquement les paramètres fournis à la requête en fonction du résultat de la requête précédente.

La principale difficulté concernant cette récupération de données a été la limite de requête. L'API limitant à une trentaine de requête par seconde, et ne retournant les chansons que 200 par 200, nous avons implémenté dans un premier temps un délai entre chaque période de requête pour éviter les requêtes inutiles pendant le timeout.

Nous avons ensuite implémenté un système pour permettre de définir quelles parties du dataset télécharger, en définissant le numéro de la première et de la dernière chanson souhaitée, ainsi qu'un système de sauvegarde automatique, qui tire avantage des temps d'attente à cause du timeout pour enregistrer les données. Toutes ces mesures nous ont permis de répartir le téléchargement entre nos machines pour ensuite récupérer les données, en pré filtrant déjà les colonnes que nous savions inutiles à nos visualisations, comme les url par exemple.

## Traitement des données

Pour commencer le traitement des données, nous avons étudié la base de données WASABI, et nous avons listé tous les champs dont nous aurions besoin pour nos représentations :

CHANSONS
- *releaseDate*
- *title*
- *publicationDate*
- *genre*
- *rank*
- *explicitLyrics*

ALBUMS
- *X_id..oid*
- *id_artist*
- *country*
- *language*
- *deezerFans*

ARTISTES
- *X_id..oid*
- *gender (male/female/other/"")*
- *type*
- *members*
- *name*
- *location.country*

Nous avons ensuite effectué tous les traitements nécessaires pour rendre les données exploitables en R. Nous nous sommes d'abord rendus compte que la base de données contenait énormément de champs vides, et de nombreuses incohérences au niveau du format des données. Notre première action a été de filtrer les fichiers csv, pour ne garder que les colonnes citées ci-dessus, afin d'éviter de conserver des informations inutiles qui alourdissent les traitements.

Nous avons commencé par supprimer les rares chansons sans titre, en considérant qu'il s'agissait d'erreurs de saisie. Ensuite, nous avons ajouté un champ avec une année associée à chaque chanson en fonction des dates de publications et de sortie de chaque chanson.

Pour obtenir une année exploitable, nous avons commencé par standardiser le format des dates dans les colonnes *releaseDate* et *publicationDate*. Nous avons supprimés les caractères *"*, *[*, *]*, et *‘* qui n'étaient pas nécessaires puis nous avons passés toutes les dates au format ISO 8601 (YYYY-mm-DD) lorsque cela était possible, et en attribuant *NA*, qui correspond à la valeur nulle en R, aux chansons

dont l'année n'était pas définie. Une fois cela fait, nous avons extrait uniquement l'année des dates formatées, puisque le reste des dates ne nous intéressent pas.

Lors de l'extraction, nous nous sommes rendues compte que certaines dates avaient été mal saisies, car les années de publications étaient supérieures à 2023, comme 2048 ou 2100. Nous avons donc corrigé ces dates à l'aide d'une table de correspondance entre les dates mal saisies et les dates réelles, que nous avons construit en vérifiant les dates de sorties des chansons. Cela nous a permis de corriger 2100, qui correspondait à 2010, ou 2048 qui correspondait à 2018. Nous avons ensuite rempli la nouvelle colonne *year*, en prenant la valeur dans *publicationDate* si elle était définie, ou dans *releaseDate* à défaut. Toutes les valeurs non définies ont simplement été remplies avec *NA,* pour faciliter les traitements futurs. Nous avons ensuite fini cette partie du traitement en supprimant ces deux colonnes qui n'étaient plus nécessaires.

Nous avons ensuite dû trouver un moyen de rendre utilisable le genre d'une chanson. En effet, c'est une information cruciale que nous utilisons dans 3 de nos visualisations. Ces genres étaient stockés sous la forme d'une liste, ce qui les rendait impossibles à utiliser en l'état.

Pour les rendre utilisables, la première chose a été de fusionner les listes de tous les genres inscrits dans le jeu de données. Pour cela, nous avons retiré tous les caractères parasites, puisque certaines listes ont été écrites avec des guillemets autour des mots, que certaines listes comportent des crochets en début et en fin de liste, alors que d'autres listes ne comportent pas ces éléments. Une fois les listes standardisées, nous les avons fusionnées, et nous avons affiché le nombre d'occurrences de chaque genre dans un histogramme.

Le but de l'histogramme, qui est situé dans le dossier *visualisation/public/distrib* était de visualiser les genres les plus représentés, ce qui permet de constater que la distribution des genres était très inégale. Nous avons choisi de conserver les 20 genres les plus fréquents sur 656, car ces 20 genres représentent plus de 75% des genres présents en fréquence d'apparition. Nous avons ensuite regroupé les différentes variations de genres, en créant 20 listes correspondant aux genres majoritaires. Par exemple, les genres 'Soft Rock','Hard Rock' et 'Folk Rock' ont été placés dans la liste 'Rock', tandis que les genres 'Jazz standard','Jazz Fusion' et 'Smooth Jazz' ont été placés dans la liste 'Jazz'. Tous les genres qui ne pouvaient pas être regroupés ont été placés dans une liste 'Other', qui permet d'identifier que la chanson a un genre renseigné qui est minoritaire comme 'Garage' ou 'Adult Contemporary Music'.

Une fois cette liste de genre établie, nous avons pu parcourir le genre pour lui assigner le genre majoritaire qui correspond au genre de sa liste. Si une chanson appartient à plusieurs genres, son genre correspondra au genre le plus représenté dans le jeu de données. Pour les chansons qui n'ont pas de genre définies, nous avons créé la catégorie 'Undefined'. Ce traitement nous a permis d'obtenir un genre par chanson, et de réduire significativement les genres, de 656 à 22, ce qui rend les visualisations basées sur les genres possibles, et permet d'éviter les genres dupliqués ou les variantes trop peu significatives.

# Parallel set : Ambre Correia

## Traitement des données

Tout d'abord, le fichier *parallel_set_preprocess.R* s'occupe du pré-traitement des données. Dans un premier temps, les colonnes utiles ont été extraites des fichier songs_cleaned_date_genre (*explicitLytrics*, *artist_name*, *id_album*, *genre*), de artists.csv (*name*, *type*, *gender*) , et albums.csv (*X_id…oid*, *explicitLyrics*).

Pour plus de clarté, *X_id…oid* a été renommé en *id_album*, et *explicitLyrics* en *explicitAlbum*. De même, le nom des artistes a été renommé en *artist_name* pour effectuer la jointure plus facilement.

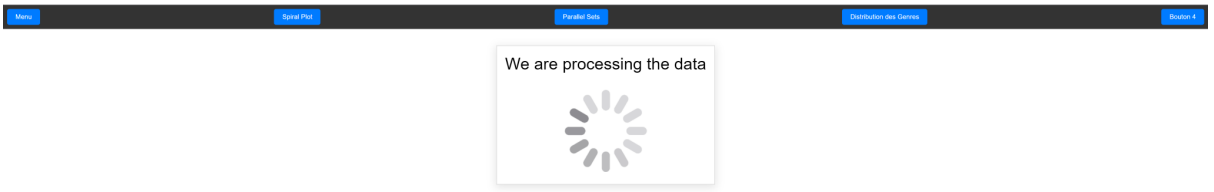
Concernant la jointure, elle a été effectuée entre les musiques et les artistes sur *artist_name*, et sur *id_album* pour les album. Les colonnes servant pour la jointure ont ensuite été supprimées, car elles n'étaient plus utiles.

Ensuite, le traitement principal consiste à supprimer les lignes vides, ou celles qui avaient des valeurs manquantes dans *type*, *gender* et *explicitAlbum*, car les données seraient inutilisables. Après cette première étape, les valeurs manquantes ont été complétées selon le processus suivant :

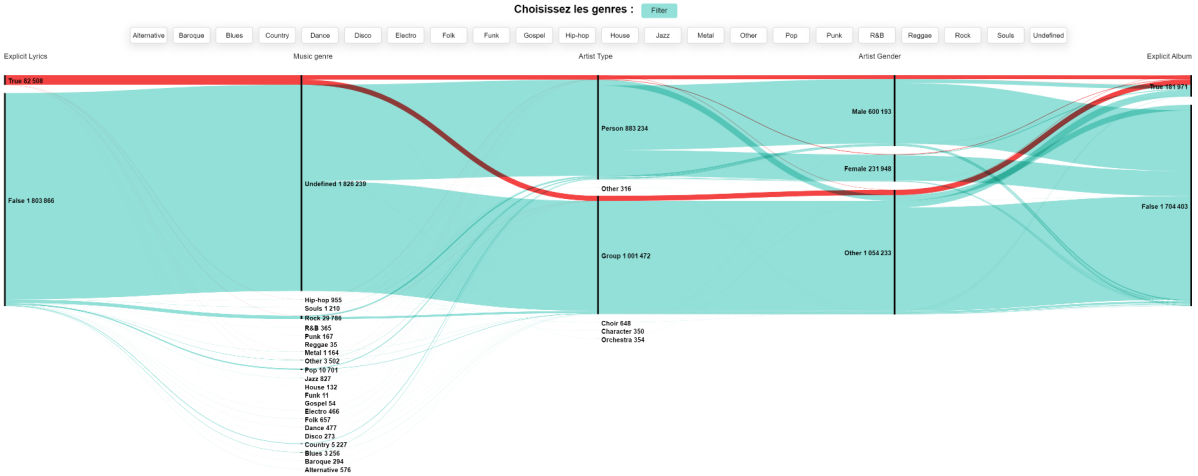
- Si le *gender* est à *male* ou *female*, alors le *type* est complété par *person*, sinon par *group*
- Si l'album contient une musique explicit, alors il est également explicit, sinon il est à *false*
- *Gender* est complété par *other*

## Visualisation

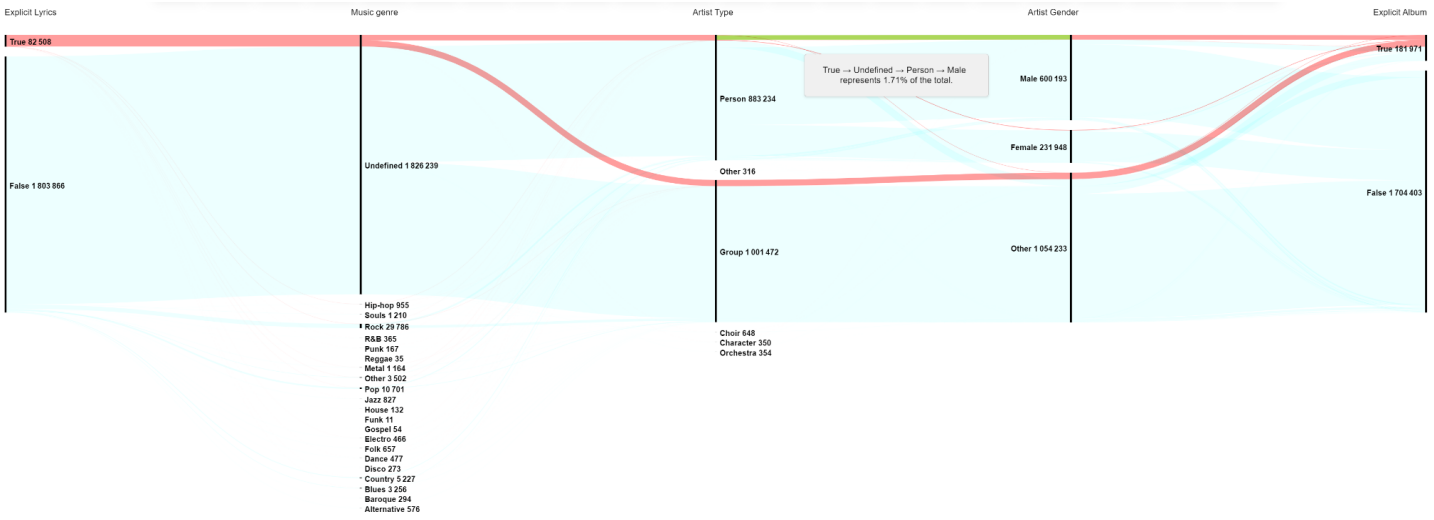
Quand nous arrivons sur la page du parallel sets, un message nous informe que les données sont en train d'être traitées, comme montré dans l'image ci-dessous. En effet, les fichiers csv fait près de 2 millions de lignes et a besoin d'un petit délai pour être traité.



Pour les deux niveaux de visualisation, nous avons tout d'abord le graphe, qui est affiché avec la couleur rouge pour les *explicitLyrics*, et bleu pour le reste. Les chiffres associés à chaque section du graphe sont également affichés de base pour plus de clarté.

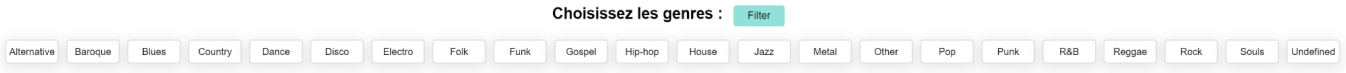



Lorsque l'on passe sur le graphe, un tooltip s'affiche en hover et il contient le chemin exact ainsi que le nombre d'occurence. Donc, par exemple, *male* affiche 600 193 nombre d'occurrence, mais le tooltip du path *True -> Undefined -> Person -> Male* va afficher 32 295, ce qui est une information différente. Enfin, lorsque l'on clique sur une section du graphe, cette section est mise en valeur et une popup s'affiche, donnant le chemin exact des informations et le pourcentage que cette section représente.

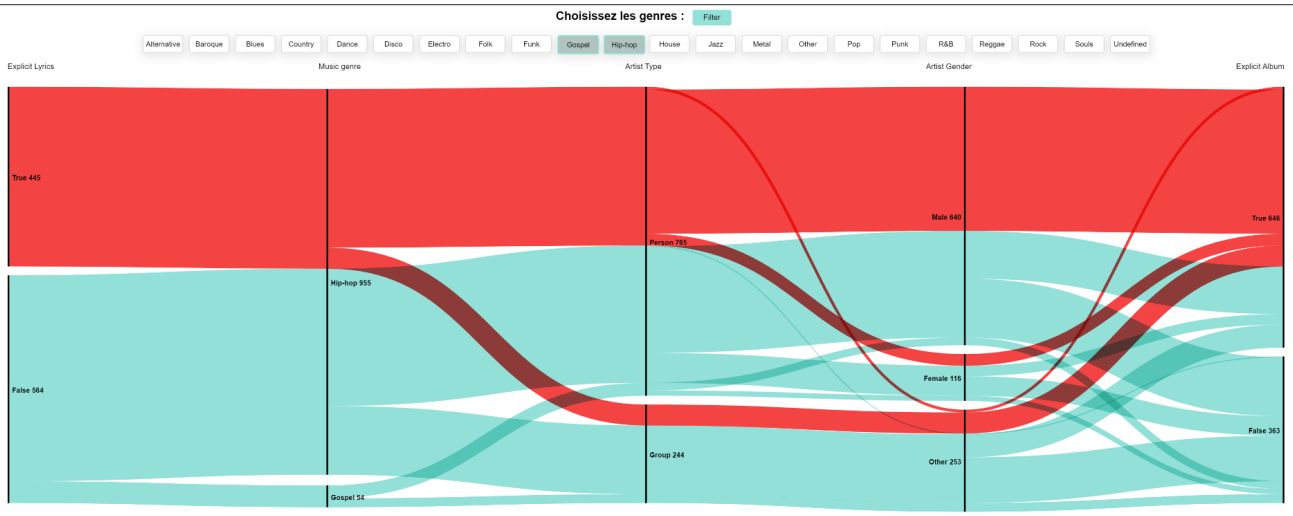



La popup qui s'affiche peut être déplacée à l'aide de la souris pour ne pas gêner la visualisation. Pour la faire disparaître, il suffit de re-cliquer sur la section de graphe sélectionnée pour revenir à l'état normal.

Pour la tâche interactive, nous retrouvons un filtre, en haut de la page, dans lequel il est possible de sélectionner le ou les genres que l'on veut filtrer. Pour cela, il suffit de les choisir puis d'appuyer sur *filtrer*. A nouveau, un message de chargement des données sera affiché à l'utilisateur.




On peut retrouver ci-dessous un exemple du graphe une fois qu'un filtre est appliqué.



## Spiral Plot : Théo Jeannes

Ma représentation consiste à afficher le nombre de chansons par genre à travers les années, en différenciant le caractère explicite de ces chansons, afin de mettre en évidence d'éventuelles tendances dans l'évolution des chansons explicites à travers les genres et les années.

### Traitements des données

Toutes les fonctions ayant servi à préparer les données du spiral plot, ou graphique en spirale, sont dans le fichier *spiral_plot_preprocess.R*. A partir des données préparées en amont, je n'ai eu besoin que des données des chansons.

La première étape a été de filtrer les chansons pour ne garder que les années, les genres et le caractère explicite ou non d'une chanson. J'ai ensuite enlevé toutes les lignes sans année, puisque ma visualisation ne permet pas de représenter ces lignes. J'ai également limité le jeu de données aux années entre 1962 et 2017, les années en dehors de cette période ne comportant que moins d'une dizaine de chansons pour certaines, et pas du tout pour le reste des années.

J'ai fini par associer à chaque ligne une valeur correspondant à son nombre d'occurrences dans mon jeu de données, pour pouvoir mesurer le nombre de chansons par année par genre et par caractère explicite ou non. Les données finales ont donc quatres colonnes, l'année *year*, le genre musical *genre,* le caractère explicite *explicitLyrics* et le nombre de chansons dans cette catégorie, *value*. Ces données sont contenues dans le fichier *spiral_plot_count.csv*.

### Visualisation

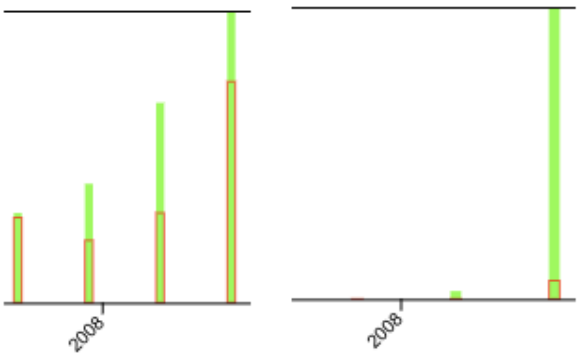
Pour ma visualisation, je suis parti du travail d'Arpit Narechania (Annexe 2), qui implémente un spiral plot. La première chose a été de remplacer les données factices par mes propres données.

Pour faire apparaître les données sur l'axe, j'ai dû adapter les échelles. La position de chaque barre sur la spirale était définie uniquement en fonction de la date. Avec mes données, cela aurait amené tous les genres à se superposer plutôt qu'à être placé côte à côte. Pour éviter cela, une map associant le genre à un nombre entre 0 et 1 est calculée à chaque affichage des données. La position correspond maintenant à l'année sommée avec la valeur du genre dans la map. Cette technique m'a permis de positionner tous les genres d'une même année côte à côte.

Pour identifier plus facilement que des genres font partie d'une même année, les genres sont regroupés par groupe en fonction de leur année, et chaque groupe se voit attribuer une couleur. J'ai choisi seulement 6 couleurs, pour ne pas surcharger visuellement le graphique. La palette de couleurs a été générée avec I Want

Hue(Annexe 3), ce qui permet d'obtenir des couleurs les plus différentes possibles, afin de réduire les chances que l'utilisateur puisse confondre deux couleurs.

J'ai ensuite fait le choix d'afficher le nombre de chansons, soit la hauteur de chaque barre sur une échelle logarithmique. Sans celle-ci, avec une échelle linéaire, le genre 'Undefined' qui représente la vaste majorité de chaque année aurait été si grand que les autres barres n'auraient été affichées que sur cinq ou dix pour cent de la hauteur. L'inconvénient de cette échelle est sa difficulté de compréhension, les comparaisons étant bien plus faciles sur une échelle linéaire. Cependant au vu de la place limitée sur l'écran, une échelle logarithmique permet d'afficher des différences importantes facilement, ce qui est le but recherché puisque nous voulons identifier des tendances.



Echelle logarithmique et linéaire pour le même échantillon

Pour identifier les chansons à caractères explicite, j'ai choisi de les encadrer en rouge, couleur qui contraste avec toutes les couleurs présentes dans la palette, et de diminuer légèrement l'opacité. Cela permet de distinguer facilement si un genre sur une année comporte plus de chansons explicites que de chansons implicites.

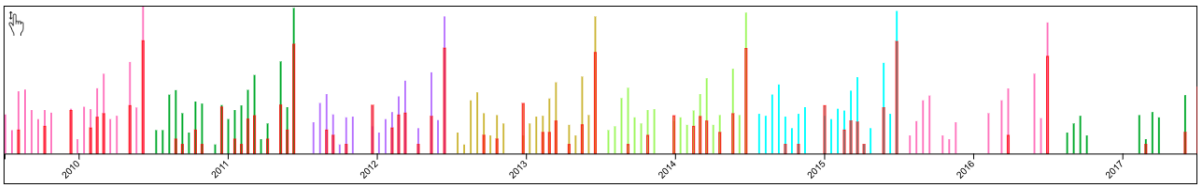

J'ai par la suite modifié le tooltip déjà existant, pour faire correspondre les labels avec nos données, et pour ajouter des informations sur la proportion. En effet, puisque la comparaison avec une échelle logarithmique n'est pas intuitive, j'ai ajouté une tooltip, affichée lorsque que l'utilisateur survole une barre, qui contient à la fois le nombre de chansons de la catégorie et le nombre de chansons pour ce genre de cette année, mais également un pourcentage représentant cette valeur, pour faciliter les comparaisons entre les années.


J'ai ensuite ajouté plusieurs filtres différents, qui permettent de restreindre les données affichées pour faciliter l'exploration, ou pour n'afficher que les données souhaitées si l'utilisateur sait déjà ce qu'il cherche et souhaite simplement localiser l'information. Le premier filtre est un double slider qui permet de définir la période à afficher sur le graphique. J'ai ajouté deux champs pour saisir la date manuellement si l'utilisateur trouve cette saisie plus confortable. Il

y a ensuite une liste des différents genres, pour permettre à l'utilisateur de choisir les genres affichés, en bleu des genres qui ne doivent pas l'être, en blanc. Pour finir, deux cases à cocher permettent de filtrer les chansons par caractère explicite ou non, si l'utilisateur ne souhaite comparer que le nombre de chansons explicites en fonction des années par exemple. Le bouton *Filter*, tout en bas du panneau avec les filtres, permet de valider les changements, pour afficher les données filtrées.

L'étape suivante a été d'adapter dynamiquement la taille à la fois du graphique, et la largeur de chaque barre. Pour adapter la taille du graphique et du panneau de filtres, j'ai simplement défini des proportions en CSS, pour rendre l'affichage responsive. La largeur de chaque barre est quant à elle calculée dynamiquement en fonction du nombre d'informations à afficher, afin de s'assurer que deux barres ne rentrent pas en collision, ce qui créerait une occlusion non désirée.

La dernière chose à faire a été d'ajouter un deuxième niveau de visualisation, pour pouvoir voir les proportions, et la répartition entre explicite et implicite correctement, sans perdre l'information globale. Pour cela, j'ai introduit un histogramme sous le spiral plot, pour permettre une visualisation globale, et de zoomer sur une partie spécifique du spiral plot en même temps.




Cet histogramme reprend les mêmes règles d'affichage que le spiral plot, et permet de zoomer ou de dézoomer à l'aide de la molette, comme indiqué par l'icône en haut à gauche du graphique, pour afficher plus ou moins d'informations. Lorsque l'utilisateur clique sur une barre, que ce soit dans le spiral plot ou dans l'histogramme, les données affichées dans l'histogramme se centre instantanément sur la barre cible, ce qui permet à l'utilisateur de visualiser les proportions du genre et de la période choisie, et son voisinage proche, tout en gardant une vue globale dans le spiral plot, ce qui permet de remettre les détails dans le contexte.

J'ai du pour finir adapter les labels, pour placer l'année au milieu de son genre, et modifier la position de la tooltip par rapport au curseur, afin de la placer au-dessus du curseur, pour ne pas cacher la visualisation.

# Annexes

**Annexe 1:**

- https://github.com/JeannesTheo/infovis : Github avec le code source
- https://drive.google.com/drive/folders/1KMQ-7-d675h42WQPgVOmDHN4jC1Lqb27?usp=sharing : Intégralité des fichiers csv utilisés durant le projet.

**Annexe 2:**

- https://gist.github.com/arpitnarechania/027e163073864ef2ac4ceb5c2c0bf616

Spiral Plot créé par Arpit Narechania, avec un tooltip et des données factices.

**Annexe 3:**

- https://medialab.github.io/iwanthue/

Site permettant de choisir une palette de couleur optimales, ou chaque couleur est le plus éloigné possibles des autres couleurs