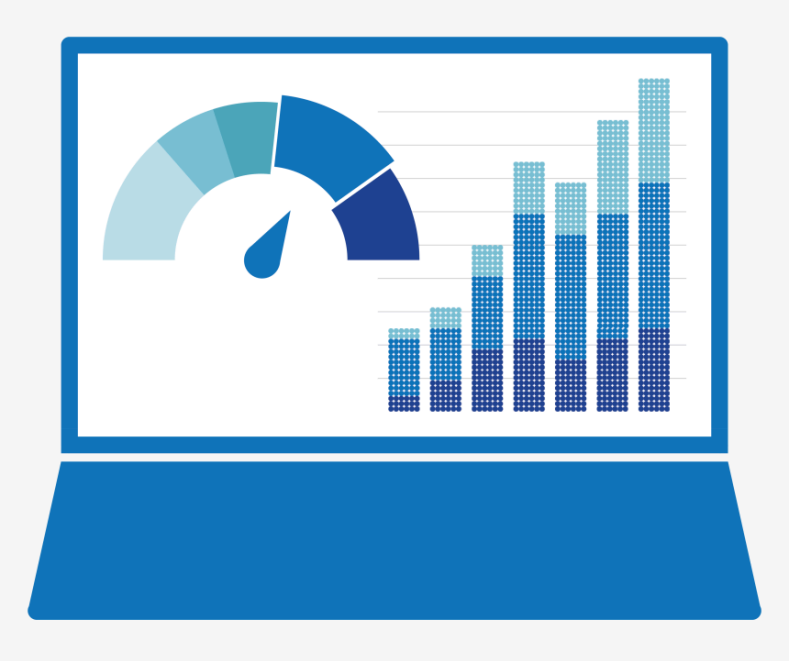
UNIVERSITÉ CÔTE D'AZUR

RAPPORT DE PROJET

« Projet Visualisation des données »



Les sources du projet sont disponibles sur GitHub

Réalisé par :

Benjamin VALLEIX
Augustin GIRAUDIER
Quentin ESCOBAR
Guillaume WAUQUIER
Khalil BOUSSIK

Université Côte d'Azur - Année 2024-2025

SOMMAIRE

I – Introduction

1 . Contexte

Avec la montée en puissance des plateformes de streaming musical, Spotify s'est imposé comme l'un des acteurs majeurs du secteur. Chaque jour, des millions de morceaux sont écoutés à travers le monde, générant une quantité colossale de données. Ces informations peuvent révéler des tendances musicales, des préférences d'écoute selon les périodes, ou encore mettre en lumière les artistes et chansons les plus populaires.L'objectif de notre projet est de transformer ces données brutes en visualisations interactives et accessibles, permettant de comprendre l'évolution des streams, de découvrir les morceaux et artistes en vogue, et d'analyser le comportement des utilisateurs au fil du temps.

2 . Utilisateurs ciblés

Notre projet s'adresse à plusieurs types d'utilisateurs :

- **Professionnels de l'industrie musicale** : pour analyser les performances des morceaux et des artistes ou identifier les tendances émergentes
- **Artistes et producteurs** : pour comprendre leur audience, suivre l'évolution des écoutes de leurs chansons
- **Chercheurs et universitaires** : Pour étudier les phénomènes culturels à travers la consommation musicale

3 . Technos utilisés

Pour réaliser ce projet, nous avons exploité plusieurs technologies :

- **Python** avec différentes librairies de traitement de données
  - **Pandas** : Manipulation et traitement des données en DataFrame.
  - **NumPy** : Manipulation de tableaux de données.
- **Plotly** pour créer des graphiques interactifs et dynamiques
- **Dash** (framework basé sur Plotly) pour le déploiement d'un tableau de bord interactif

II – Jeux de données

1. Explication du dataset choisi

Dans le cadre de ce projet, nous avons utilisé deux jeux de données afin d'obtenir le maximum de paramètres pour réaliser nos visualisations. Chacun des datasets n'ont pas le même nombre de paramètres, ou du moins les mêmes intitulés, de plus certains paramètres nous sont inutiles et doivent être supprimés. C'est pour cela que nous avons dû réaliser une phase de filtrage et fusionnement afin d'exploiter un seul et unique dataset nettoyé avec les champs qui nous intéressent.

Voici un tableau récapitulatif des deux datasets utilisés :

	Dataset A spotify-most-streamed-songs	Dataset B music-artists-popularity
Nombres d'entrées	1 000 (taille réduite)	1 466 083 (taille importante)
Objectifs	Récupérer les musiques les plus populaires de Spotify	Permet d'ajouter plus de contexte sur nos artistes et musique (genre, pays) Filtrer le très grand nombre de données non utilisées
Exemple de formats	track_name, artist(s)_name,artist_count released_year,released_month,released_day, in_apple_playlists, energy_%	mbid,artist_mb, artist_lastfm, country_mb, country_lastfm, tags_mb tags_lastfm, listeners_lastfm, scrobbles_lastfm, ambiguous_artist

Le premier dataset est centré sur le top de Spotify, et nous donne des informations sur chaque morceau. On y trouve le nom du son, les artistes, la date de sortie, le nombre de playlists où la chanson apparaît (que ce soit sur Spotify, Apple Music ou Deezer), et le nombre de streams. Ce qui rend ce dataset particulièrement intéressant, ce sont les caractéristiques qu'il contient, le tempo (BPM), la tonalité, mais aussi des scores plus techniques comme la danceability ou l'énergie du morceau.

Le second dataset vient enrichir ces informations avec un angle différent. Il se concentre sur les artistes eux-mêmes, avec leur pays d'origine, les genres

musicaux auxquels ils sont associés, et même leur popularité sur d'autres plateformes comme Last FM. Cette base de données est beaucoup plus large, ce qui dans notre cas n'est pas forcément bon car sur le million d'artistes de ce dataset seulement 1 000 sont présents dans le dataset Spotify .

2 . Filtrage des données

Le filtrage des données constitue une étape essentielle dans tout projet de visualisation ou d'analyse des données. Il vise à assurer la qualité, la cohérence et la pertinence des informations utilisées. Voici un résumé des différentes étapes de filtrage et de prétraitement effectuées sur notre dataset :

Chargement des données

Les données brutes ont été importées depuis un fichier CSV en utilisant la bibliothèque Pandas. Le dataset initial contenait un total de 954 lignes et 29 colonnes.

```python
df = pd.read_csv("./dataset/dataset.csv")
```

Analyse des valeurs manquantes

Une première analyse a été réalisée pour détecter les colonnes comportant des valeurs manquantes. Le but était d'évaluer l'impact de ces valeurs manquantes sur la qualité du dataset. Nous avons par exemple remarqué qu'une grande partie des pays (plus de ⅔) étaient manquantes. L'objectif final était d'effectuer des traitements spécifiques, si nécessaires, pour gérer ces valeurs manquantes après la fusion des deux datasets. (ex. : suppression, imputations, remplissage des valeurs grâce à une moyenne).

```python
missing_values = df.isnull().sum()
```

Renommage des colonnes

Pour faciliter la manipulation des données et améliorer la lisibilité du code, les colonnes du dataset ont été renommées. Ce renommage a également permis

d'assurer une uniformité dans les noms, en supprimant les caractères spéciaux et en remplaçant certains termes peu explicites. Par exemple :

- track_name a été renommé en track,
- artist.s._name a été renommé en artist_name.

```python
df.rename(columns=renommer_colonnes, inplace=True)
```

Cela a simplifié la compréhension et l'utilisation des colonnes dans les étapes suivantes du projet.

Suppression des colonnes inutiles

Certaines colonnes, jugées inutiles pour l'analyse et la visualisation des données, ont été supprimées. Ces colonnes incluent :

- artist_mb,
- artist_lastfm,
- country_lastfm.

La suppression de ces colonnes a permis d'alléger le dataset et de se concentrer uniquement sur les données pertinentes.

```python
df.drop(columns=['artist_mb','artist_lastfm','country_lastfm'],
inplace=True)
```

Gestion des doublons

Une analyse des doublons a été effectuée pour identifier les enregistrements redondants. Un total de 3 doublons a été détecté et supprimé afin de garantir l'unicité des enregistrements et d'éviter des biais dans les analyses ultérieures.

```python
duplicates_before = df.duplicated().sum()
df = df.drop_duplicates()
```

Cette méthode permet de traiter les doublons parfaits (c'est à dire que toute la ligne est identique à une autre), nous avons également procédé à un second

traitement des doublons en nous basant sur le nom des titres musicaux. Ce traitement a été réalisé à la main en filtrant les champs track_name.

Normalisation des données

Afin d'uniformiser les valeurs textuelles, certaines colonnes ont été normalisées en convertissant leur contenu en minuscules et en supprimant les espaces superflus. Les colonnes suivantes ont été concernées par cette étape :

- track,
- artist_name, artist_count
- mode, key
- country_mb,
- tags_lastfm.

Cette étape a permis de garantir que les valeurs comparables soient traitées de manière homogène, éliminant ainsi les erreurs dues à des variations de casse ou à des espaces inattendus.

```python
df[col] = df[col].astype(str).str.strip().str.lower()
```

Sauvegarde du dataset filtré

Enfin, le dataset filtré et nettoyé a été sauvegardé dans un nouveau fichier CSV intitulé *dataset_filtered.csv*. Cela garantit la traçabilité du travail effectué et facilite son utilisation pour les étapes ultérieures du projet.

```python
df.to_csv("./dataset/dataset_filtered.csv", index=False)
```

Ces étapes de filtrage et de prétraitement ont permis de transformer un dataset brut en un jeu de données cohérent, prêt pour les étapes d'analyse et de visualisation avec Plotly. Chaque étape nous a permis d'améliorer la qualité des données, tout en éliminant les obstacles potentiels à une interprétation fiable des résultats. Le fichier de filtrage python est disponible sur notre GitHub, ainsi que le script R qui a permis de fusionner les deux datasets.

III – Techniques de visualisation

1. **Technique 1** (ESCOBAR Quentin)

A - Introduction

Cet objectif vise à créer une visualisation interactive des collaborations musicales à travers une approche basée sur des graphes. L'objectif principal est de transformer des données de streaming en une représentation visuelle permettant d'analyser les relations entre artistes.

B - Caractéristiques des données d'entrées

Cet objectif s'appuie sur trois paramètres du dataset. Le nombre de streams qui permet de quantifier la popularité d'une chanson. Cette métrique influence la taille des nœuds dans le graphe et l'intensité des connexions.

Ensuite, nous utilisons les noms des artistes, qui peuvent apparaître soit comme artiste principal, soit comme collaborateur. La gestion des collaborations nécessite un traitement particulier car les mentions de featuring peuvent prendre différentes formes ("feat.", "ft.", "&", etc.).
Enfin, le titre de la chanson nous permet d'identifier de manière unique chaque collaboration et d'enrichir les informations affichées lors de l'exploration du graphe.

C - Techniques utilisées

Le graph a été créé avec NetworkX. Les artistes sont représentés par des nœuds et leurs collaborations par des arêtes. Cette structure permet de visualiser les relations entre artistes.

Afin d'obtenir une visualisation claire et exploitable, nous avons choisi de nous concentrer sur les artistes dépassant un certain seuil de streams. Prendre en compte l'ensemble des artistes rendrait le graphe illisible avec des milliers de nœuds et d'arêtes s'entrecroisant, masquant ainsi les patterns de collaboration importants.

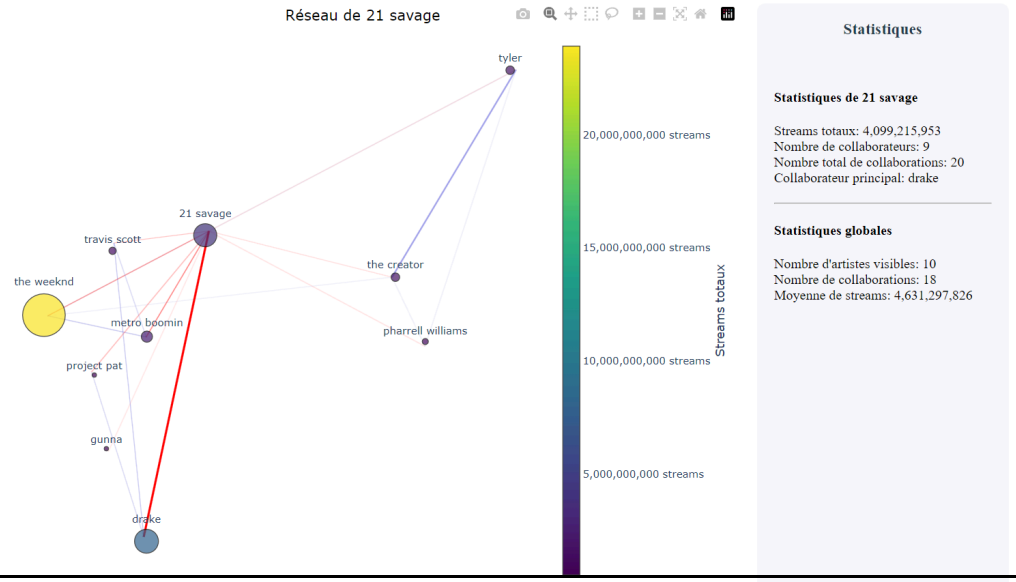
Le cœur de notre implémentation repose sur la construction d'un graphe pondéré :

```python
build_graph(df):
    # Identification des artistes les plus streamés
    artist_streams_total =
df.explode('artists_list').groupby('artists_list')['streams'].sum()
    stream_threshold = artist_streams_total.quantile(0.5)

    # Construction des relations entre artistes
    G = nx.Graph()
    for idx, row in df_filtered.iterrows():
        for pair in combinations(row['artists_list'], 2):
            if G.has_edge(pair[0], pair[1]):
                G[pair[0]][pair[1]]['streams'] += row['streams']
            else:
                G.add_edge(pair[0], pair[1], streams=row['streams'])
```

L'interface Dash nous a permis d'ajouter une barre de recherche pour sélectionner un artiste spécifique et d'afficher son réseau de collaborations, accompagné de statistiques comme son nombre total de streams et ses principaux collaborateurs. Un slider de filtrage par nombre de streams permet d'ajuster la densité du graphe

D - Résultats et déductions

On remarque que l'artiste le plus bénéfique à avoir en feat est The Weeknd. Certains artistes moins connus maintiennent des réseaux de collaboration denses et actifs et rivalisent avec de plus grands artistes.



2. **Technique 2** (VALLEIX Benjamin)

A - Introduction

Dans cette visualisation, nous nous concentrons sur l'analyse des flux des caractéristiques musicales au fil des années. Le but est de visualiser l'évolution de la dansabilité (incitation à danser), de l'énergie (intensité musical), de l'acoustique et de la balance (harmonie des différents éléments musicaux) des morceaux afin d'identifier des périodes stratégiques pour la sortie de nouvelles musiques.

B - Techniques utilisées

Nous avons décidés d'utiliser deux techniques particulières afin d'obtenir un meilleur rendu final pour cette visualisation :

- Graphes en aires de flux

Les caractéristiques musicales sont représentées sous forme de couches empilées pour visualiser leur évolution. Chaque courbe représente une caractéristique spécifique : dansabilité, énergie, acoustique et balance. Les aires permettent de mettre en évidence les tendances musicales mais aussi l'importance (relative) de chaque caractéristique. Pour cela nous avons utilisé `Scatter` (lib Plotly) avec l'option `stackgroup` pour empiler les caractéristiques.

```python
fig.add_trace(go.Scatter(
    x=filtered_df['released_year'], y=filtered_df['danceability'],
    mode='lines', stackgroup='one', name='Dansabilité',
line=dict(color='blue')
))
```

Par exemple, la méthode add_trace permet d'ajouter une nouvelle trace au graphique existant. Chaque trace représente une caractéristique musicale, et les traces superposées permettent d'obtenir notre graphique en aire de flux.

- Point de moyenne mensuelles

Nous avons fait le choix d'ajouter des points afin de représenter la moyenne mensuelle de chaque caractéristique, ce qui permet une granularité supplémentaire. Cette approche met en évidence des variations saisonnières ou des anomalies spécifiques.

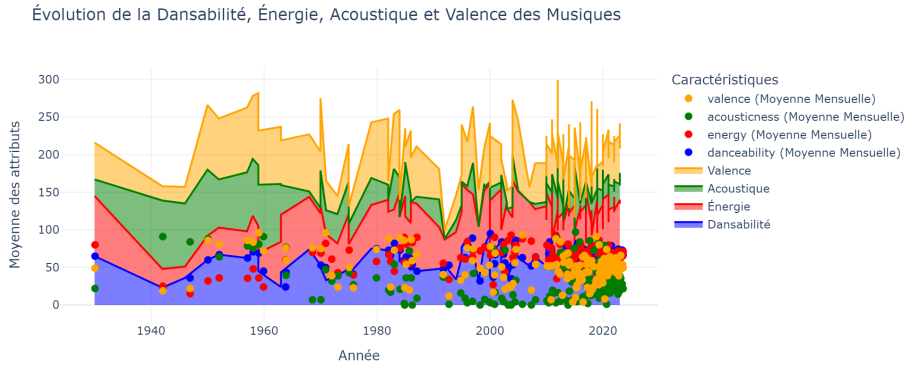
C - Interactions et explorations

Le fait d'utiliser un composant Dash dcc.RangeSlider a permis de filtrer les données affichées en fonction d'une période choisie par l'utilisateur

```python
dcc.RangeSlider(
    id='year-slider',
    min=df_agg['released_year'].min(),
    max=df_agg['released_year'].max(),
    . . .
)
```

Par exemple, en déplaçant le curseur, on peut observer l'évolution des caractéristiques musicales pour une période spécifique. Cela nous as été utile pour se concentrer sur les années 80 ou après 2000, périodes que l'on a trouvé intéressante à analyser pour leurs caractéristiques musicales

D - Résultats et déductions



Les résultats obtenus grâce à cette visualisation révèlent des tendances musicales marquantes sur plusieurs décennies :

- La dansabilité montre une tendance globale à la hausse, en particulier après les années 2000 (évolution vers des rythmes plus dansants, hipop)
- Les niveaux d'énergie ont des fluctuations notables, avec des pics dans les années 80 et une légère baisse après 2010.

L'ajout de points pour les moyennes mensuelles a permis notamment d'identifier certaines variations saisonnières. Par exemple, les morceaux plus "énergiques" semblent plus fréquents en été, tandis que ceux à forte acoustique augmentent en fin d'année, probablement en raison des genres associés à des fêtes.

3. **Technique 3** (BOUSSIK Khalil)

A - Introduction

Dans cette visualisation, nous voulons voir la corrélation entre le nombre de playlists Spotify et le nombre de streams, en tenant compte de l'année de sortie des titres. L'objectif est de comprendre comment l'intégration dans des playlists influence la popularité d'une chanson au fil du temps.

B - Techniques utilisées

Cet objectif repose sur un scatter plot interactif. Le graphique est construit en plaçant chaque chanson selon ses caractéristiques :

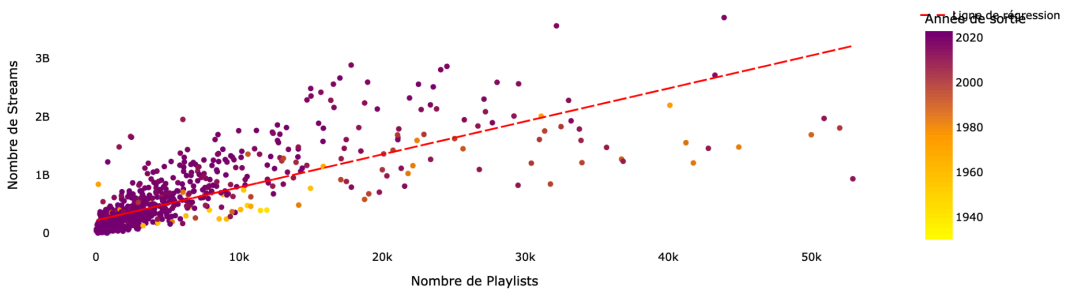
```python
fig = px.scatter(
    data_filtered,
    x='spotify_playlists',
    y='streams',
    color='released_year'
)
```

C - Interactions et Explorations

L'interface permet plusieurs niveaux d'exploration :

- Survol des points pour accéder aux détails de chaque chanson (titre, artiste, stats)
- Échelle de couleur temporelle du jaune au violet selon l'année de sortie
- Ligne de régression rouge pointillée pour visualiser la tendance générale
- Zoom et sélection de zones spécifiques pour une analyse détaillée

D - Résultats et Déductions



Les résultats de notre visualisation montrent une corrélation positive entre le nombre de playlists et les streams. La ligne de régression confirme cette tendance générale. Néanmoins, la dispersion des points révèle des exceptions significatives : des chansons peu présentes en playlists peuvent avoir beaucoup de streams, et inversement. La distribution des couleurs par année de sortie indique que cette relation varie selon les périodes, suggérant une évolution de l'impact des playlists sur le succès des titres au fil du temps.

4. **Technique 4** (WAUQUIER Guillaume)

A - Introduction

Cet objectif vise à analyser l'évolution des streams musicaux à travers le temps, en mettant l'accent sur les variations saisonnières et l'impact des collaborations. L'objectif est de créer une visualisation interactive permettant d'explorer les tendances d'écoute à différentes échelles temporelles.

B - Caractéristiques des données d'entrée

Notre analyse s'appuie sur les métriques suivantes. Le nombre de streams, les données temporelles (année, mois, jour de sortie) qui sont essentielles pour notre analyse chronologique. Pour enrichir l'interaction, nous utilisons également le nom des artistes et des titres, ainsi que le nombre d'artistes par morceau pour évaluer l'impact des collaborations.

C - Techniques utilisées

La visualisation repose sur Plotly et utilise une approche de séries temporelles. Le prétraitement des données temporelles est crucial :

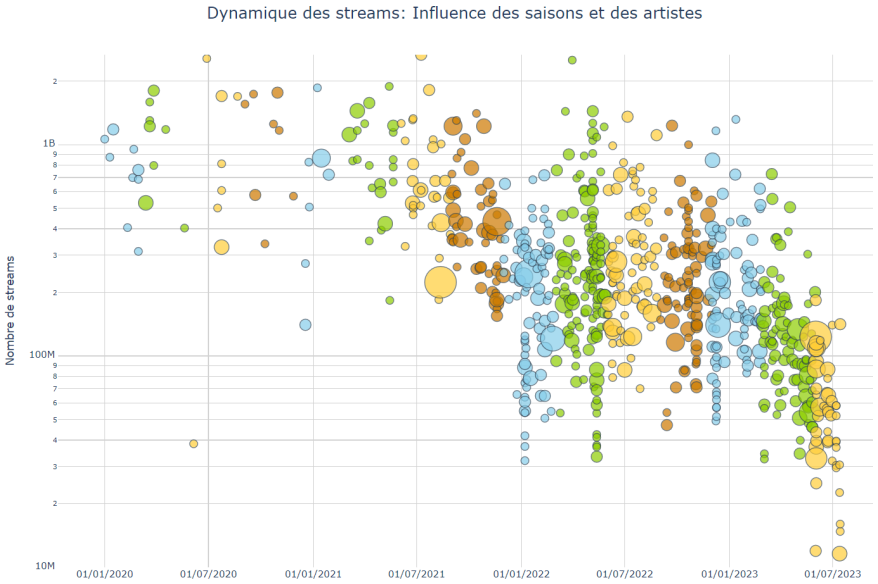
```python
# Création d'une date unique à partir des composants temporels
df['date'] = pd.to_datetime(df[['released_year', 'released_month',
'released_day']])

# Agrégation mensuelle des streams
monthly_streams = df.groupby(['date'])['streams'].sum().reset_index()

# Création du graphique interactif
fig = go.Figure()
fig.add_trace(go.Scatter(
```

```
    x=monthly_streams['date'],
    y=monthly_streams['streams'],
    mode='lines',
    text=df['trackname'] + ' - ' + df['artistname'],
    hoverinfo='text+x+y'
))
```

L'interface intègre un système de focus+context avec un range slider permettant de zoomer sur des périodes spécifiques, offrant ainsi une vue détaillée tout en conservant le contexte global des données.

D - Résultats et déductions

L'analyse révèle des pics saisonniers marqués, particulièrement en été et pendant les fêtes. Les collaborations entre artistes montrent une corrélation positive avec le nombre de streams, suggérant leur importance croissante dans le succès commercial. Cette visualisation permet aux professionnels de l'industrie d'identifier les périodes optimales pour la sortie de nouveaux titres.



5. **Technique 5** (GIRAUDIER Agustin)

A - Introduction

Cet objectif vise à explorer la relation entre le tempo (BPM) et la “liveness” des morceaux musicaux. L'objectif est de comprendre la relation entre BPM et liveness et d'identifier comment ces deux paramètres contribuent au succès des hits modernes.

B - Caractéristiques des données d'entrée

Notre analyse s'appuie sur trois métriques du dataset.
Le BPM (battements par minute) mesure le tempo d'une chanson, variant généralement de 60 à 180. La liveness, comprise entre 0 et 1, indique la probabilité qu'un morceau ait été enregistré en live. Enfin, le nombre de streams permet de quantifier le succès de chaque titre. Ces métriques combinées nous permettent d'évaluer l'impact du rythme et de l'authenticité sur la popularité.

C - Techniques utilisées

Notre visualisation s'articule autour de plusieurs composants interactifs. Un slider de sélection permet de filtrer les plages de BPM souhaitées, offrant une exploration précise des différentes catégories rythmiques :

```python
dcc.RangeSlider(
    id='bpm-slider',
    min=df['bpm'].min(),
    max=df['bpm'].max(),
    step=1,
    value=[60, 140],  # Plage par défaut
    marks={i: str(i) for i in range(60, 180, 20)}
)
```

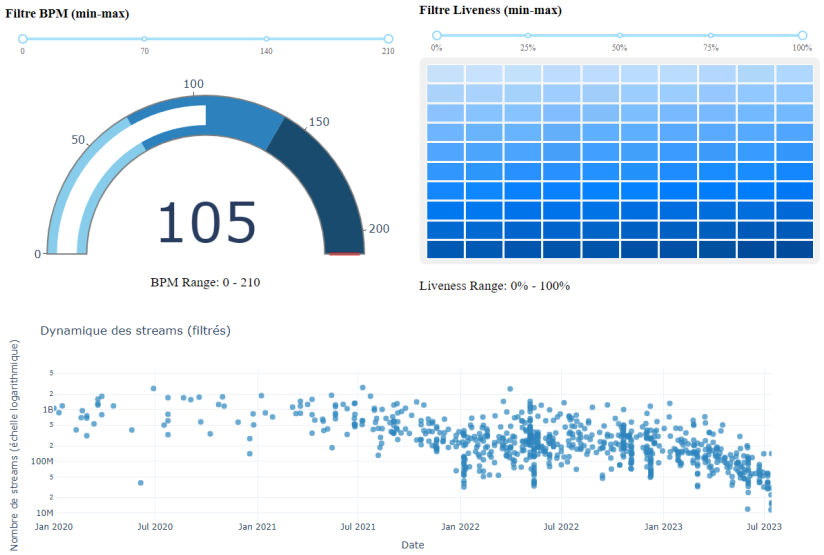
Le graphique principal utilise une méthode de visualisation par nuage de points, où chaque point représente un morceau. L'axe X représente le BPM, l'axe Y le nombre de streams, et la couleur du point indique le niveau de liveness:

```python
fig.add_trace(go.Scatter(
    x=filtered_df['bpm'],
    y=filtered_df['streams'],
    mode='markers',
    text=filtered_df['trackname'] + ' - ' + filtered_df['artistname'],
    hoverinfo='text+x+y'
))
```

Cette approche permet de visualiser simultanément les trois dimensions de nos données (BPM, streams, liveness) tout en maintenant une interface intuitive. L'interactivité du graphique permet d'explorer les détails de chaque morceau au survol, affichant le titre et l'artiste.

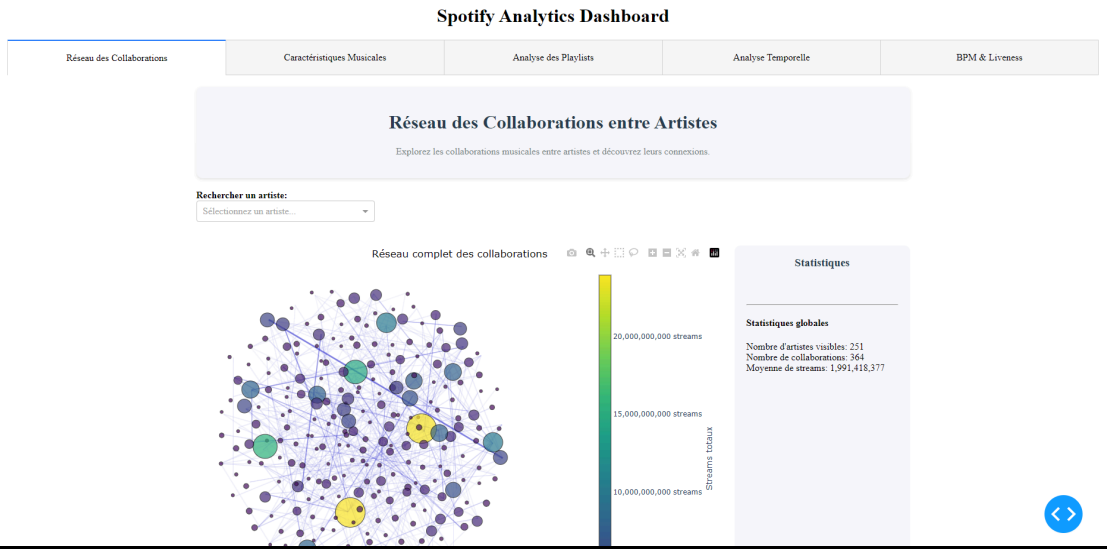
D - Résultats et déductions

L'analyse révèle une concentration de succès dans la plage 120-130 BPM, particulièrement pour les genres pop et électro. Les morceaux avec une liveness modérée (0.4-0.6) montrent généralement de meilleures performances en streaming, suggérant que les auditeurs préfèrent un équilibre entre production studio et authenticité live. Ces résultats offrent des indications précieuses pour optimiser les futures productions musicales.



6. Dashboard

Pour percevoir une vue globale de chacune des techniques de visualisation et de leurs rendus, nous avons réalisés un dashboard (script python qui se base sur nos fichiers jupyter), dont voici un extrait :



IV – Conclusion

À travers ce projet de visualisation de données Spotify, nous avons développé cinq techniques complémentaires qui offrent des perspectives différentes sur les tendances musicales et les comportements d'écoute. L'utilisation combinée de Python, Pandas, Plotly et Dash nous a permis de créer des visualisations interactives et informatives.

Nos principales réalisations incluent :

- Une visualisation des réseaux de collaboration entre artistes qui a révélé l'importance stratégique de certains artistes comme The Weeknd dans l'écosystème musical
- Une analyse temporelle des caractéristiques musicales montrant l'évolution des préférences vers des morceaux plus dansants après 2000
- Une étude de la relation entre présence dans les playlists et streams qui confirme une corrélation positive mais avec des exceptions notables
- Une analyse des variations saisonnières des streams qui met en évidence des pics pendant l'été et les fêtes
- Une exploration de la relation entre BPM et "liveness" révélant une zone optimale autour de 120-130 BPM

Ces visualisations offrent des informations précieuses pour les professionnels ou les artistes. Elles permettent de mieux comprendre les facteurs de succès d'un morceau et d'identifier des tendances qui peuvent guider les stratégies de production et de sortie musicale.