Projet Spectre GUI : Identification d'un instrument de musique à partir de métadonnées spectrales

Amine Iklil
Maxime Blanchard
Marie Harnois

diginamic FORMATION
« Votre passeport pour l'emploi numérique »

Juin 2023

Diginamic : Consultant Data analyst POEC PIC ATLAS

Introduction

Le son est une forme de donnée lorsqu'il est numérisé et stocké:

L'analyse du Big Data sur les données sonores présente plusieurs avantages et applications potentielles :

1. Reconnaissance vocale et transcription automatique
2. Surveillance et sécurité
3. Analyse musicale
4. Détection d'anomalies
5. Analyse de sentiment

Ces applications montrent comment l'analyse du Big Data sur les données sonores peut être appliquée dans divers domaines pour extraire des informations précieuses, améliorer les performances et prendre des décisions éclairées..

Problématique

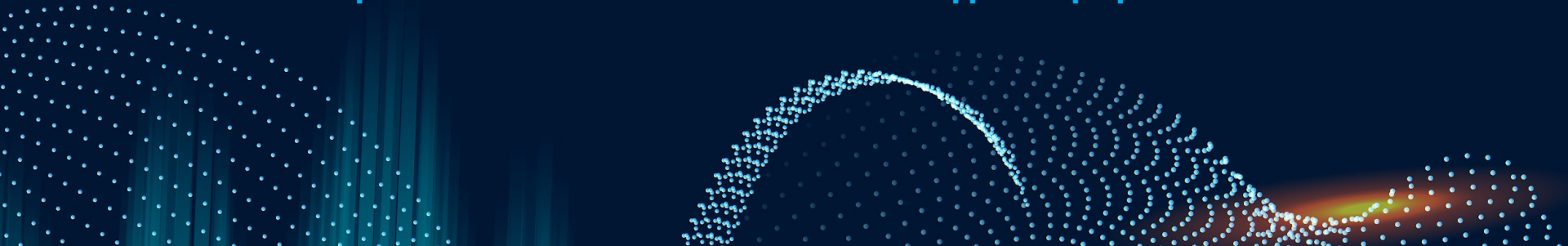
Les instruments de musique produisent des caractéristiques spectrales spécifiques qui peuvent être exploitées pour les différencier. Cependant, cette tâche présente des défis techniques importants.

Construction d'une base de données d'instruments.

Extraction des métadonnées spectrales.

Adaptation aux variations en termes de tonalité, d'articulation, de dynamique et de technique de jeu.

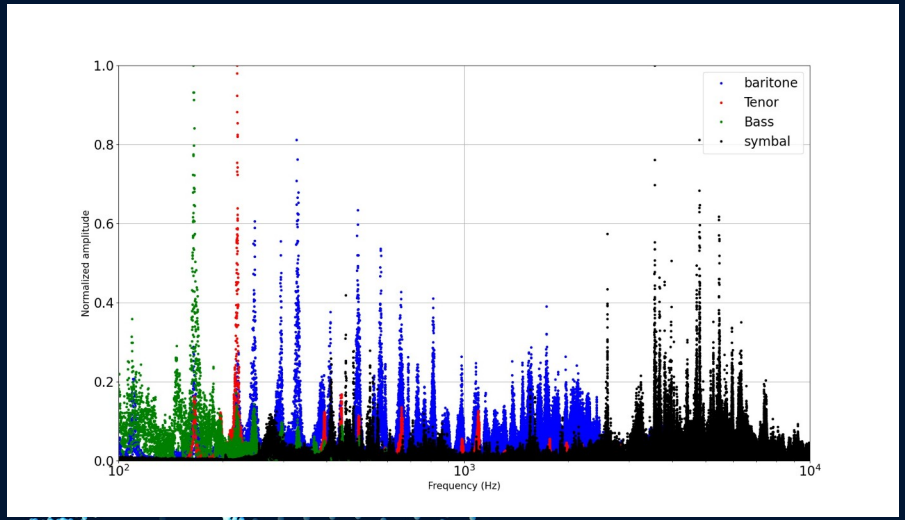
Validation expérimentale afin d’évaluer l'efficacité de l'approche proposée et des tests.



Problématique

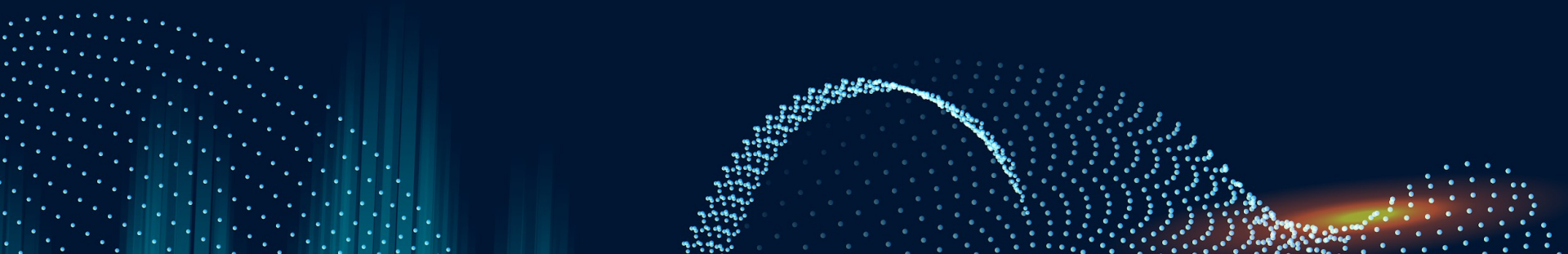
Les instruments de musique produisent des caractéristiques spectrales spécifiques qui peuvent être exploitées pour les différencier. Cependant, cette tâche présente des défis techniques importants.

Classification et apprentissage automatique :
Les métadonnées spectrales extraites des enregistrements audios doivent être utilisées pour former des modèles de classification capables de distinguer les différents instruments de musique.

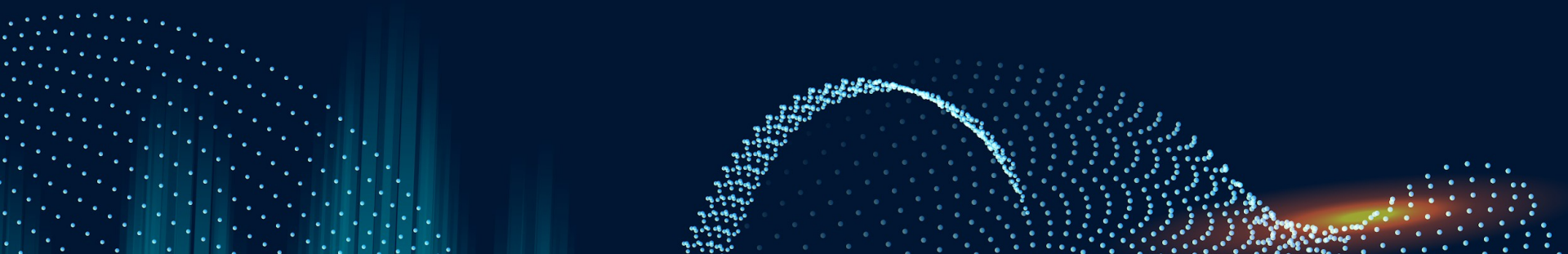


Plan de présentation

1. Analyse et import des données
2. Description des datasets
3. Traitement complexe des données
4. Interaction avec l’utilisateur – Application Projet spectre GUI



Analyse et import des données

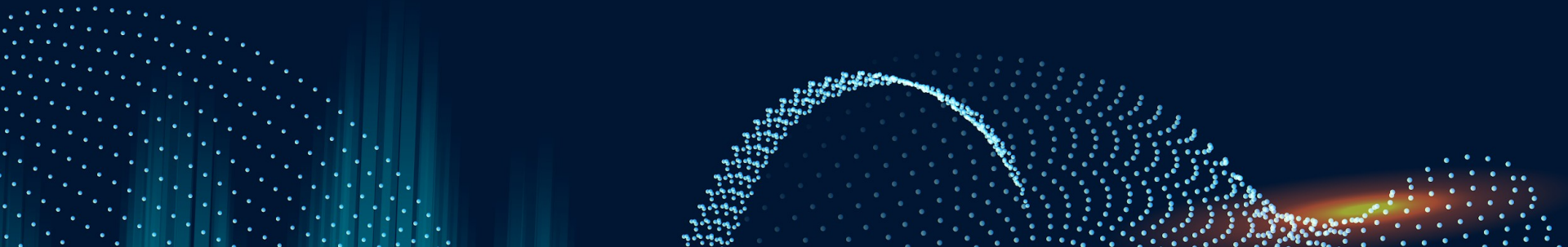


Explorations de la base de données

Le jeu de données se base sur une bibliothèque de son qui comporte des bandes de son issue d'instruments de musique de différentes catégories (https://theremin.music.uiowa.edu/).

1. **Elle contient 1304 Instruments**
2. **Chaque instrument à un potentiel d’enregistrement de 10 fichiers audio**
3. **Chaque fichier comporte potentiellement plusieurs notes**

Fonction python : Web-scraping.py permet de télécharger la base de données depuis le site



Description des fichiers audios

- **Chaque fichier comporte potentiellement plusieurs notes et informations**

Informations nominatives issues du nom du fichier

type instrument - option - Note & octave - dynamique - pitched – instrument.

Les noms des fichiers ne sont pas standardisés

Informations numériques issues de la piste audio

Signal (Donnée brut .aiff)

Nécessite un traitement du signal pour générer plus d'information sur l'enregistrement

Post traitement des fichiers audio

- **Problématique : Les noms des fichiers ne sont pas standardisés**

Fichier python : unzipper.py permet de dézipper les fichiers contenant les enregistrements de chaque instrument

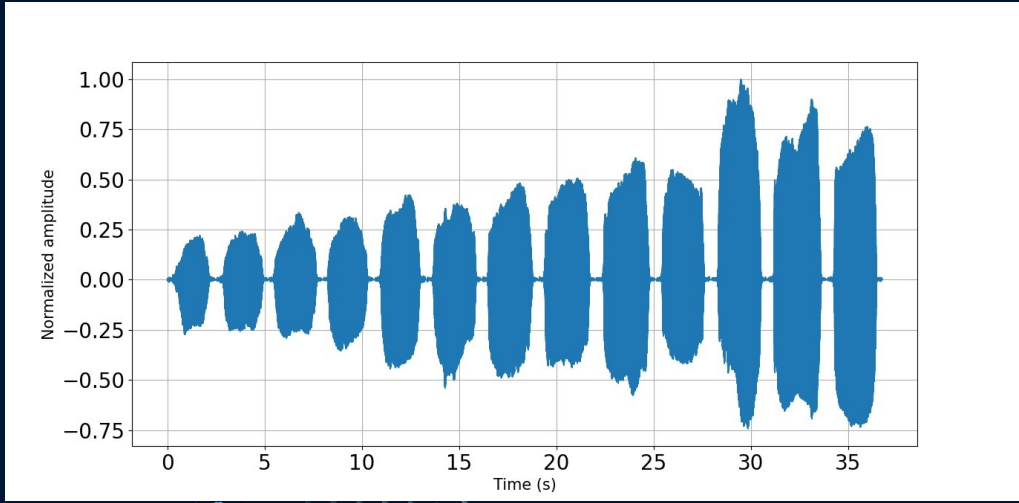
Fichier python : rename.py permet de changer les noms de tous les 500 fichiers contenants les enregistrements.

La gestion des Path est automatisé durant ce processus

Post traitement des fichiers audio

- **Problématique : Nécessite un traitement du signal pour générer plus d'information sur l'enregistrement**

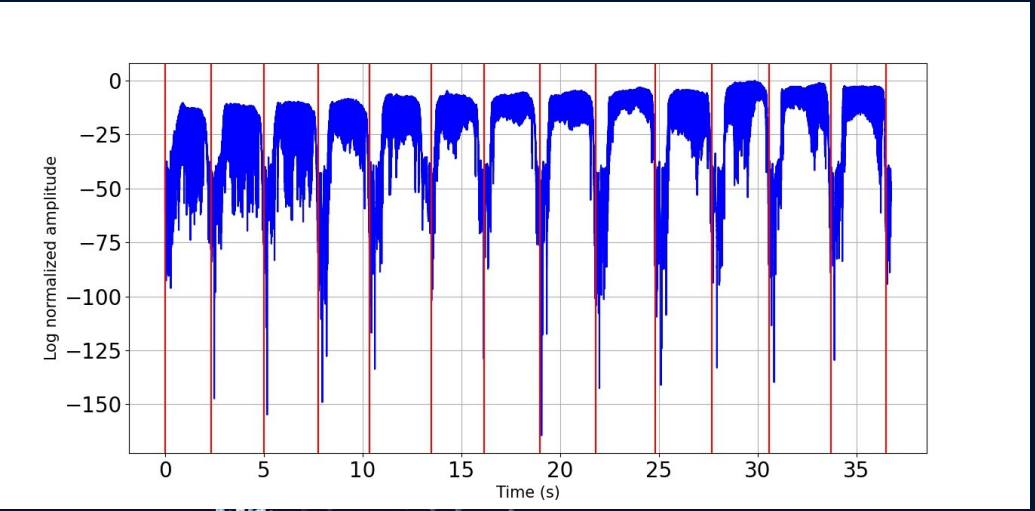
Fonction python : traitement_du_signal_spliteur permet de découper les signaux en fonction des silences



Post traitement des fichiers audio

- **Problématique : Nécessite un traitement du signal pour générer plus d'information sur l'enregistrement**

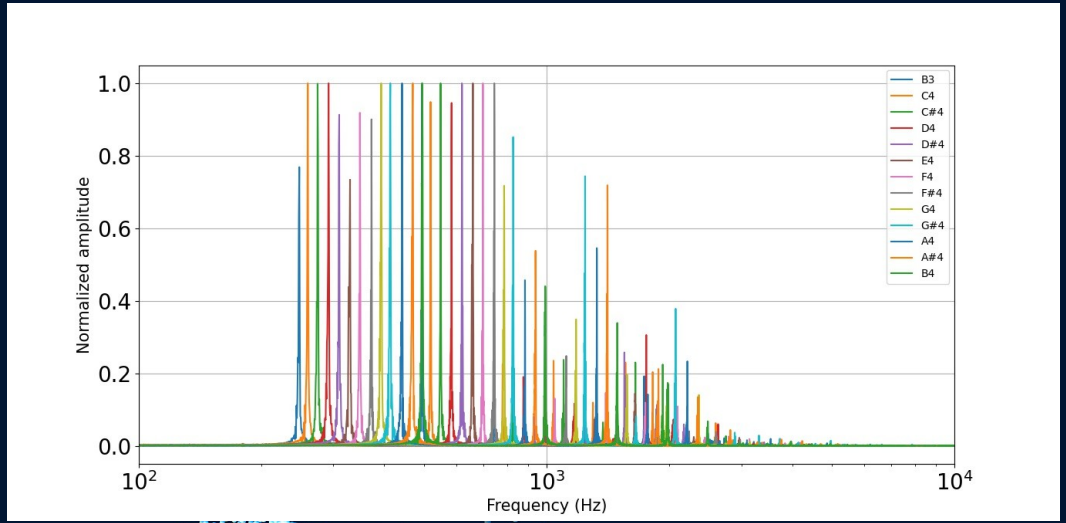
Fonction python : traitement_du_signal_spliteur permet de découper les signaux en fonction des silences



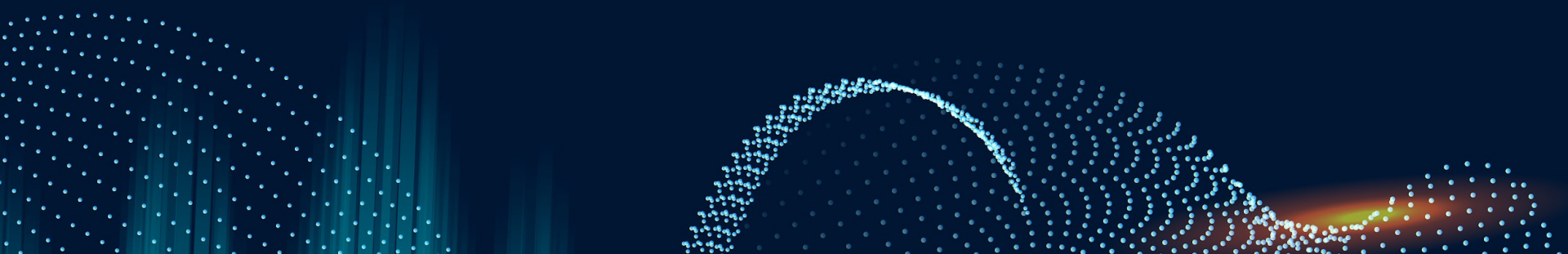
Post traitement des fichiers audio

- **Problématique : Nécessite un traitement du signal pour générer plus d'information sur l'enregistrement**

Fonction python :
parameter_extractor
permet de détecter les max des premières harmoniques et les max harmonique



Description des deux datasets



Création de la base de données

Import des données dans MongoDB

schéma validator + configuration
insertion des enregistrements et les spectres
Sous forme d'array

```
_id: ObjectId('649ac09aea0190af17b6318c')
pitched: true
type: "altoflute"
instrument: "theremin"
option: "nooption"
note: "C4"
dynamique: "ff"
▸ signal: Array
▸ spectre: Array
▸ harmonique_amplitude: Array
▸ harmonique_fondamental: Array
▸ harmonique_equ: Array
```

```
# Définir le schéma des documents de la collection
schema = {
    "$jsonSchema": {
        "bsonType": "object",
        "required": ["pitched", "type", "instrument", "option", "dynamique",
                     "fichier_octave", "signal", "spectre",
                     "harmonique_amplitude", "harmonique_fondamental",
                     "harmonique_distance_entre_harmonique",
                     "Note_first_harmonique","Note_max_harmonique"],
        "properties": {
            "pitched": {"bsonType": "bool"},
            "type": {"bsonType": "string"},
            "instrument": {"bsonType": "string"},
            "option": {"bsonType": "string"},
            "dynamique": {"bsonType": "string"},
            "fichier_octave" : {"bsonType": "string"},
            "signal": {"bsonType": "array"},
            "spectre": {"bsonType": "array"},
            "harmonique_amplitude": {"bsonType": "array"},
            "harmonique_fondamental": {"bsonType": "array"},
            "harmonique_distance_entre_harmonique": {"bsonType": "array"},
            "Note_first_harmonique": {"bsonType": "string"},
            "Note_max_harmonique" : {"bsonType": "string"},

        }
    }
}
```

Création de la base de données

Première collection

84 enregistrements
Les différentes caractéristiques de chaque document :
type instrument : altoflute, trumpet, bassflute, bbclar, flute, bassclarinet, sopsax
option : nooption, nonvib
note et octave : C#4, F4, A4, C4, D4, E4, F#4, D#4, G#4, G4, A#4, B4
dynamique : ff, mf
pitched : True
instrument : theremin
signal et spectre : informations numériques (array) issues de l'audio
12 notes pour chacun des 7 fichiers → 84 enregistrements

Création de la base de données

Deuxième collection

1304 enregistrements
Les différentes caractéristiques de chaque document :
instrument : flute, bbclar, sopsax, bassclarinet, bassarco, trumpet, altosax, ebclar, altoflute, bassflute, horn, violinarco
option : nooption, nonvib, vib, suld, sule, sula, sulg
note et octave : 0, A#2/Bb2, A#3/Bb3, A#4/Bb4, A#5/Bb5, A#6/Bb6, A#7/Bb7, A2, A3, A4, A5, A6, A7, B1, B2, B3, B4, B5, B6, B7, C#2/Db2, C#3/Db3, E2, E3, E4, E5, E6, E7, F#2/Gb2, …, G#6/Ab6, G2, G3, G4, G5, G6, G7
dynamique : ff, mf
pitched : True
type instrument : theremin
signal et spectre : informations numériques (array) issues de l'audio

Doublons dans la deuxième collection

Deuxième collection : 1304 enregistrements

```
pipeline = [
    {"$group": {
        "_id": {
            "type": "$type",
            "pitched": "$pitched",
            "option": "$option",
            "instrument": "$instrument",
            "dynamique": "$dynamique"},
        "total": {"$count": {}}}},
    {"$sort": {"total":-1}},
    {"$match": {
        "total": {"$gte":2*12}}}
]
results = collection.aggregate(pipeline)
```

instrument	nombre enreg
[bassclarinet, mf, nooption]	6.0
[bbclar, mf, nooption]	6.0
[ebclar, mf, nooption]	4.0
[flute, mf, vib]	4.0
[sopsax, mf, nonvib]	4.0
[flute, mf, nonvib]	4.0
[bbclar, ff, nooption]	3.0
[bbclar, pp, nooption]	3.0
[bassclarinet, ff, nooption]	2.0
[altoflute, mf, nooption]	2.0
[flute, ff, nonvib]	2.0
[sopsax, mf, vib]	2.0
[sopsax, pp, nonvib]	2.0
[sopsax, ff, vib]	2.0
[altoflute, ff, nooption]	2.0
[trumpet, mf, nonvib]	2.0
[flute, pp, vib]	2.0
[flute, ff, vib]	2.0
[sopsax, ff, nonvib]	2.0
[altosax, mf, vib]	2.0
[trumpet, pp, nonvib]	2.0

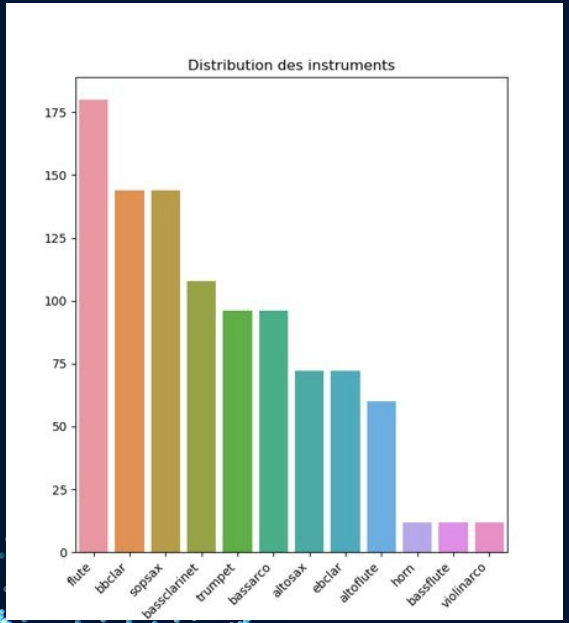
Chaque combinaison instrument/dynamique/option contient 12 enregistrements (12 notes)

Description des données

Deuxième collection : 1304 enregistrements

L’instrument le plus représenter est la flute

L’instrument le moins représenter sont le violon – basse flute - Horn

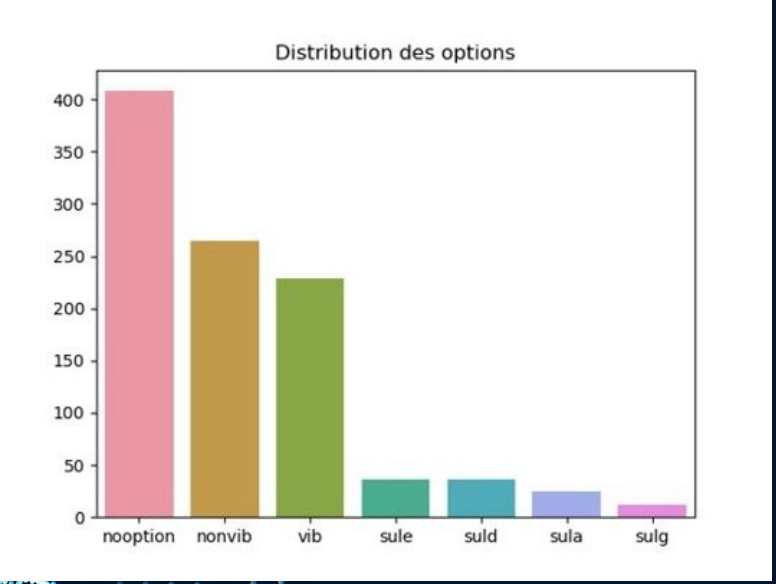


Description des données

Deuxième collection : 1304 enregistrements

L’option le plus représenter est sans option

L’option le plus représenter est sulg

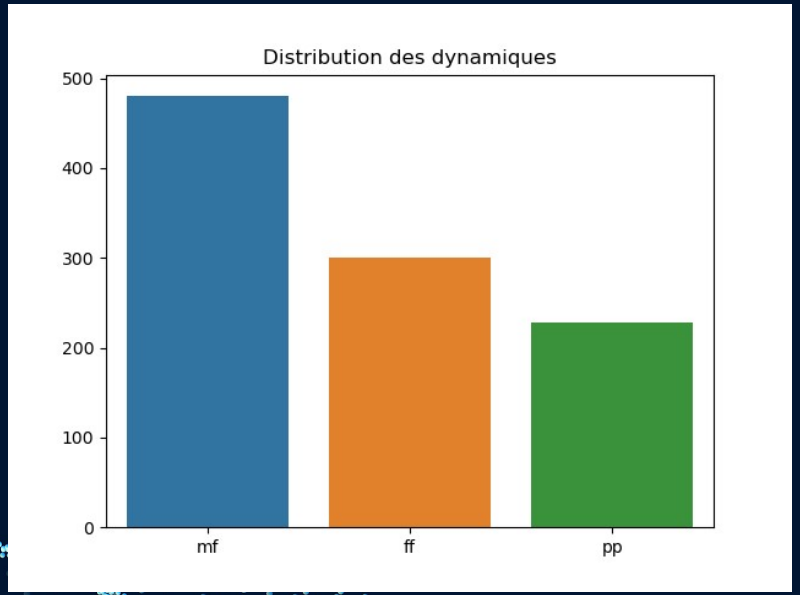


Description des données

Deuxième collection : 1304 enregistrements

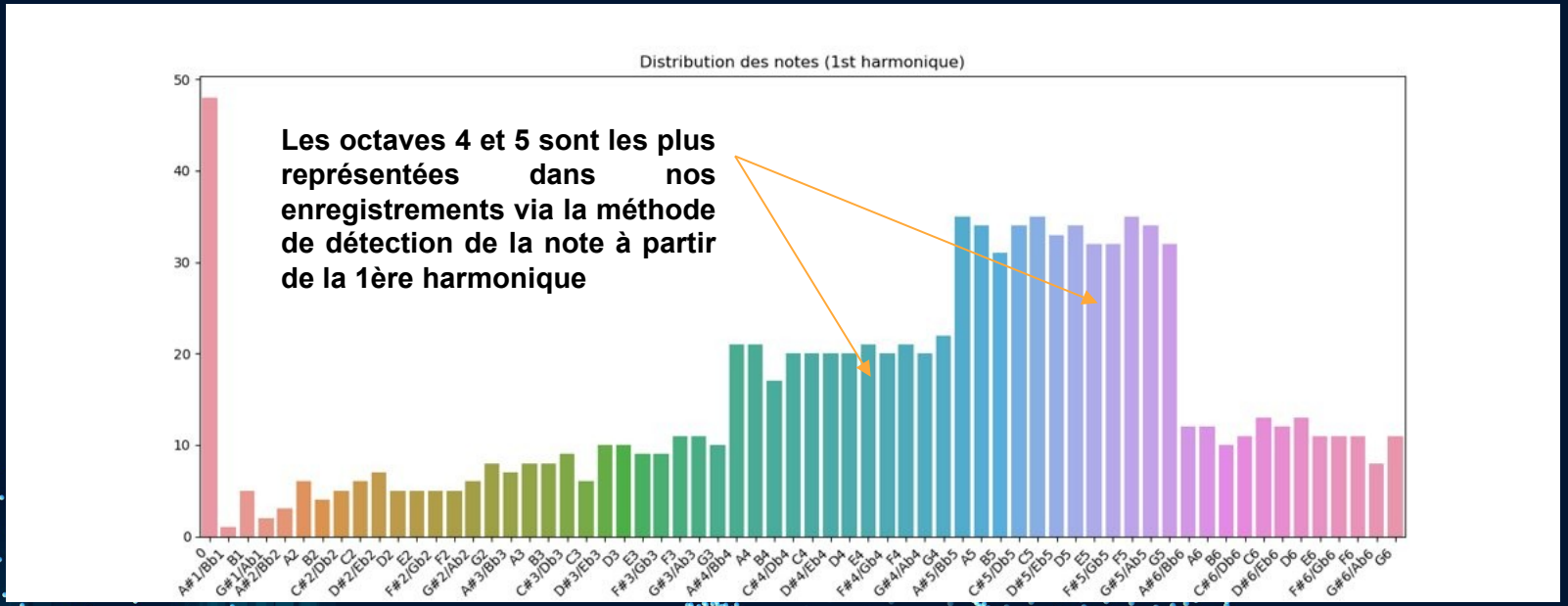
La dynamique le plus représenter pp

La dynamique le plus représenter est pp



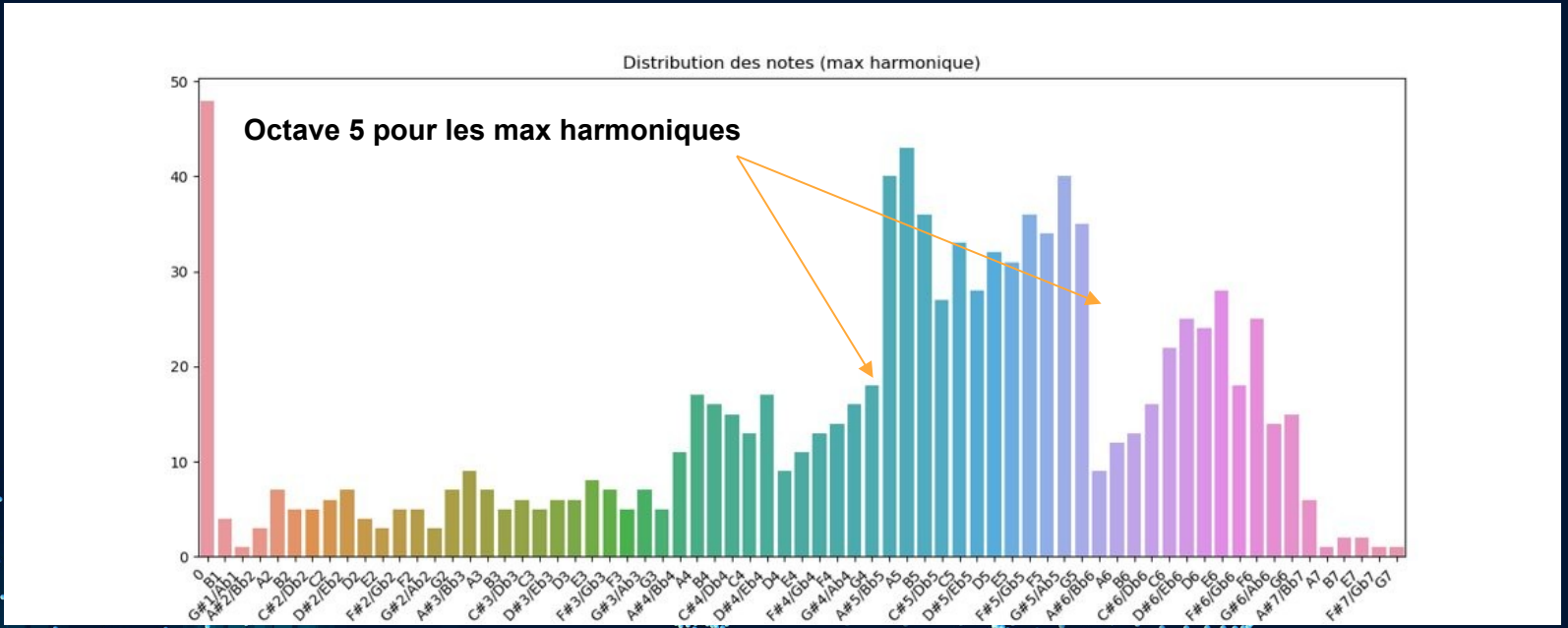
Description des données

Deuxième collection : 1304 enregistrements



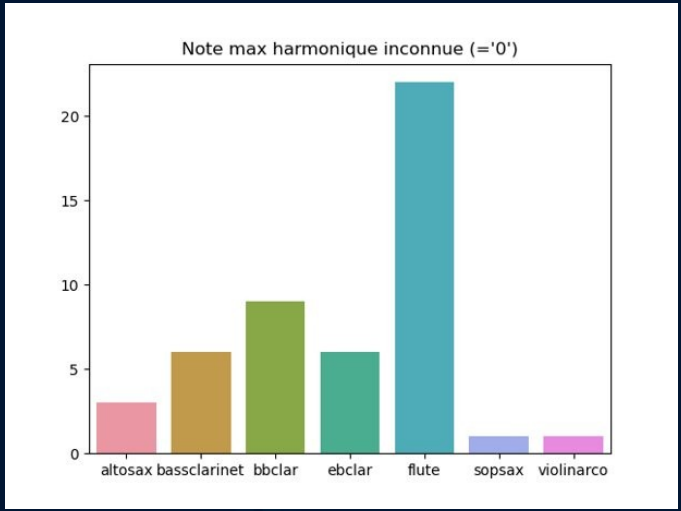
Description des données

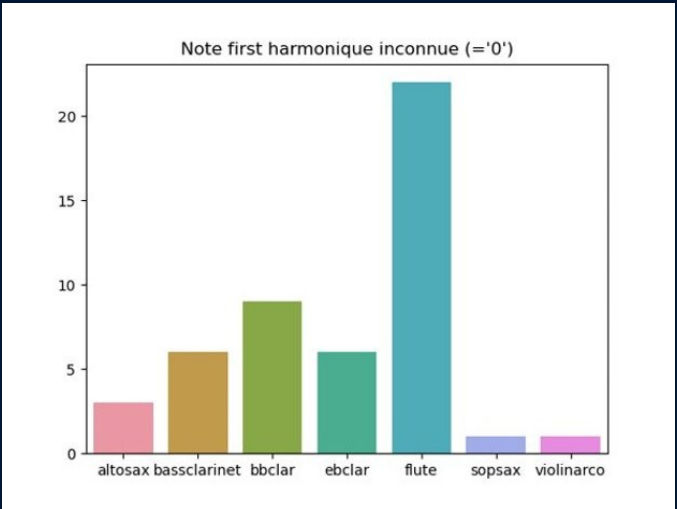
Deuxième collection : 1304 enregistrements



Notes manquantes

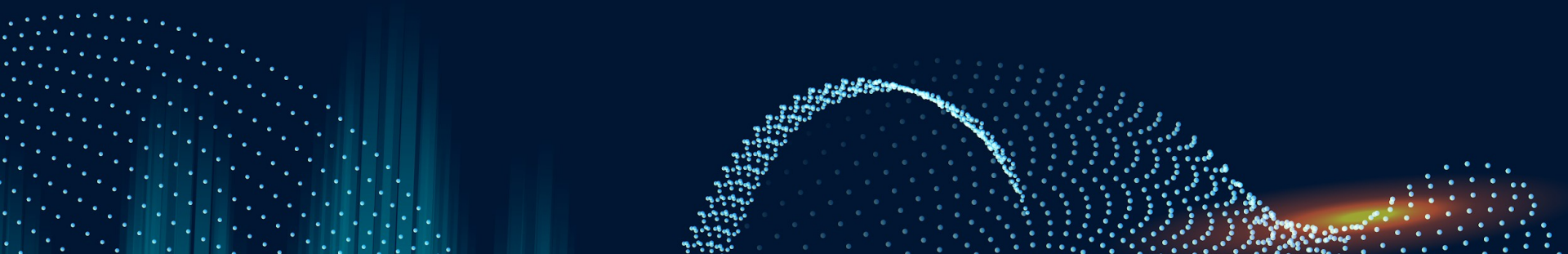
Deuxième collection : 1304 enregistrements





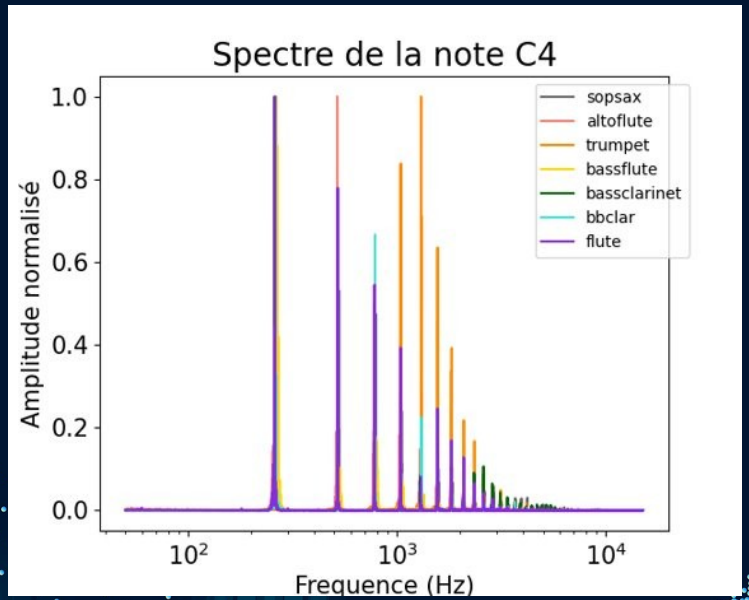

```
pipeline = [{"$match":{"Note_max_harmonique":"0"}},
            {"$group":{"_id":{
                              "note_max_harmonique": "$Note_max_harmonique",
                              "instrument" : "$instrument"
                              }, "total":{"$count":{}}}},
            {"$sort":{"_id":1}}]
results = collection.aggregate(pipeline)
```

Premiers résultats
Traitement complexe des données



Spectre de la note C4

Première collection : 84 enregistrements



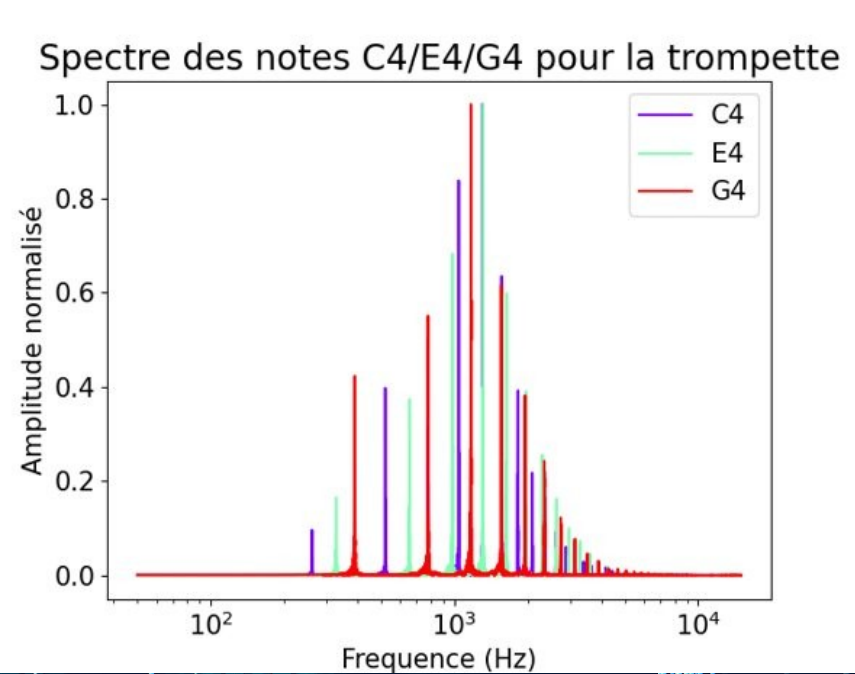
Première harmonique superposée pour tous les instruments. Changement reflétant un changement de timbre

```
liste_type_inst = []
pipeline = [{'$group':{'_id':"$type",
                       'total_instrument':
                           {'$count':{}}}},
            {'$sort':{'total_instrument':-1}}]
results = collection.aggregate(pipeline)
for res in results:
    liste_type_inst.append(res['_id'])

for i in range(len(liste_type_inst)):
    query = {"note":"C4", "type":liste_type_inst[i]}
    results = collection.find(query)
```

Spectre de la trompette pour C4 E4 et G4

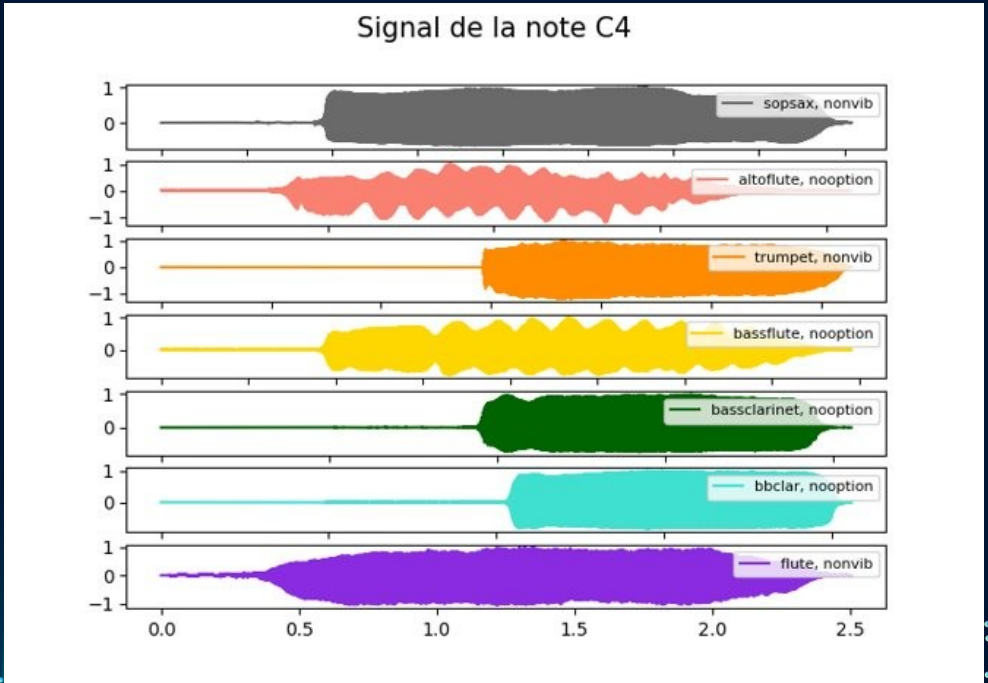
Première collection : 84 enregistrements




```
liste_note_CEG = ['C4', 'E4', 'G4']
for i in range(len(liste_note_CEG)):
    query = {"note":liste_note_CEG[i],
             "type":"trumpet"}
    results = collection.find(query)
```

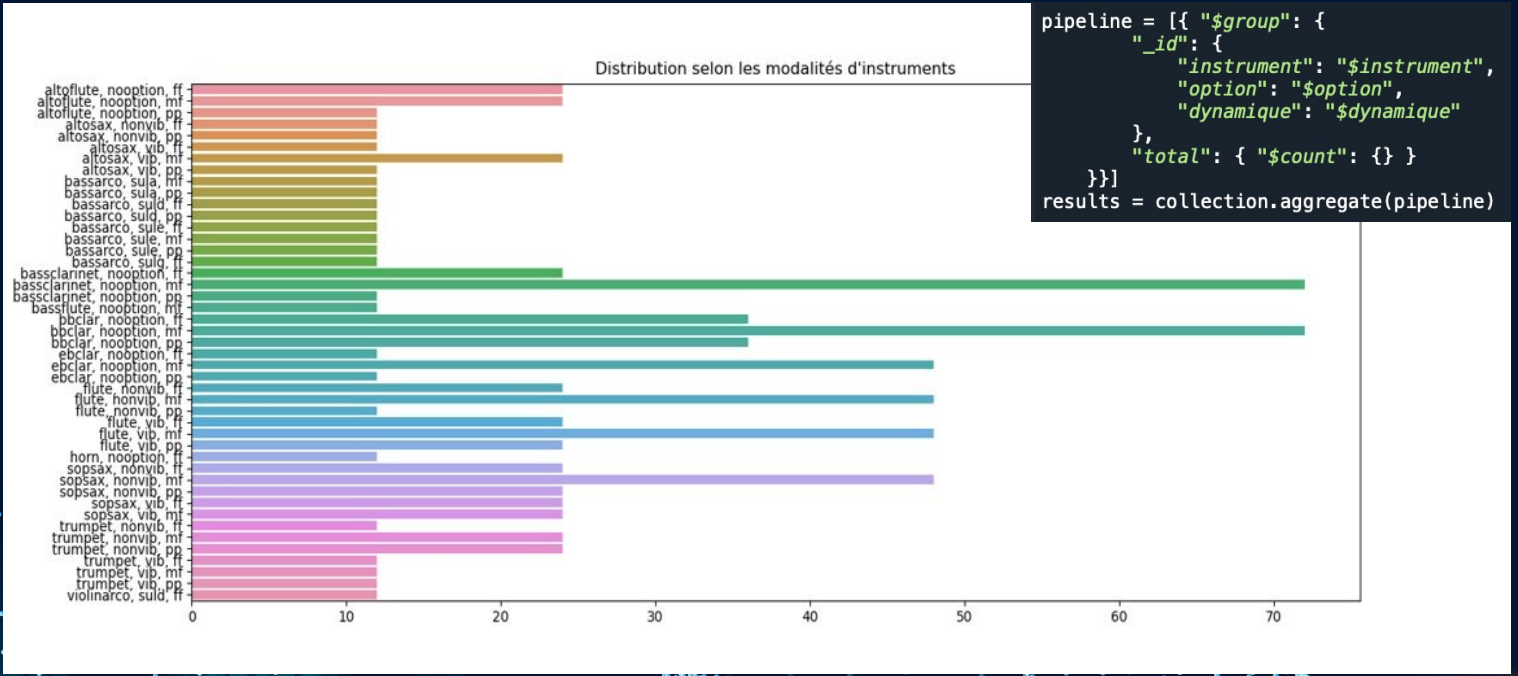
Exemple de signaux pour la note C4

Première collection : 84 enregistrements



Distribution des modalités d'instruments

Deuxiéme collection : 1304 enregistrements




```
pipeline = [{ "$group": {
        "_id": {
            "instrument": "$instrument",
            "option": "$option",
            "dynamique": "$dynamique"
        },
        "total": { "$count": {} }
    }}]
results = collection.aggregate(pipeline)
```

Requête d’agrégation

Deuxième collection : 1304 enregistrements

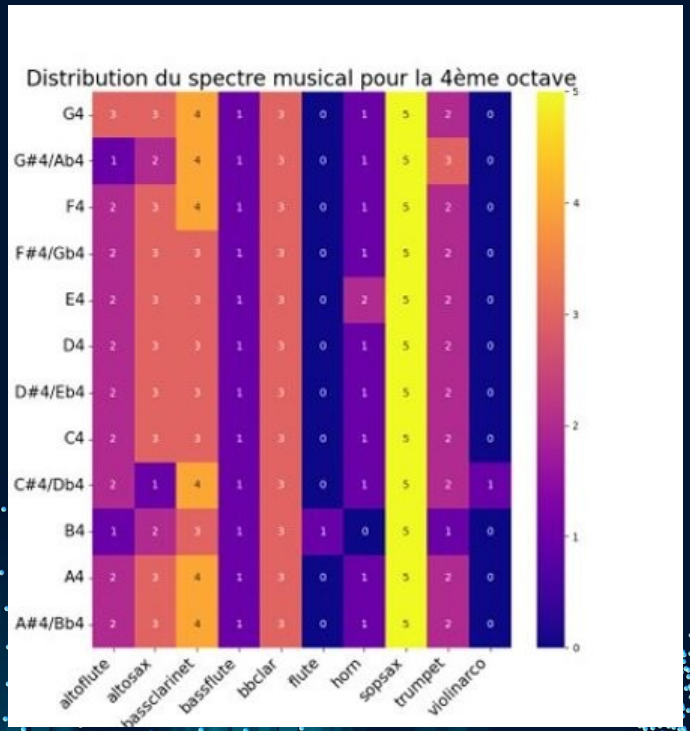
Cette requête permet d’obtenir le nombre d’enregistrements présent dans la base en fonction de l’instrument et de la note (1ère harmonique) sur l’octave 4

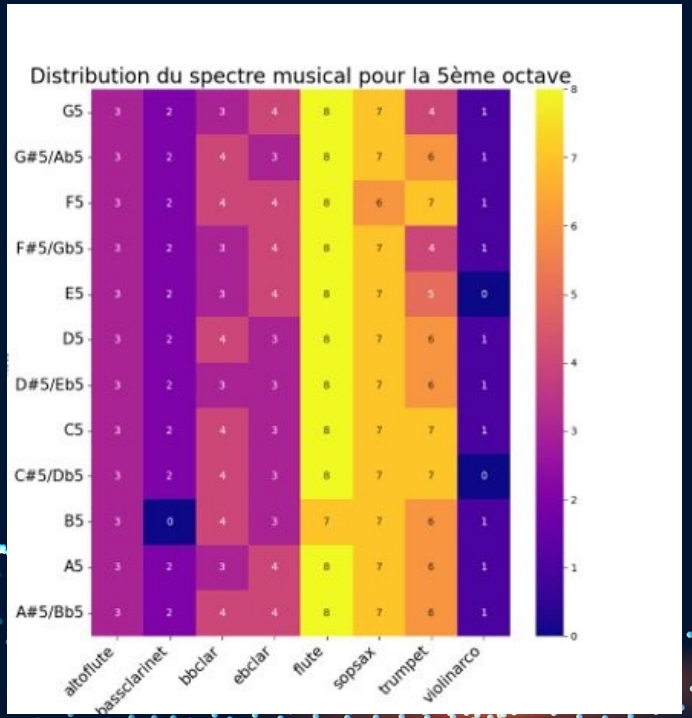
```
pipeline = [{"$match": {"Note_first_harmonique":{"$regex":"4"}}},
            { "$group": {
                "_id": {
                    "Note_first_harmonique":"$Note_first_harmonique",
                    "instrument": "$instrument"},
                "total": { "$count":{}}}},
            {'$sort':{'Note_first_harmonique':-1}}]
results = collection.aggregate(pipeline)
```

Nombre d’enregistrements par instrument/note

Deuxième collection : 1304 enregistrements

La flute est l’instrument le plus représenter sur l’octave 5





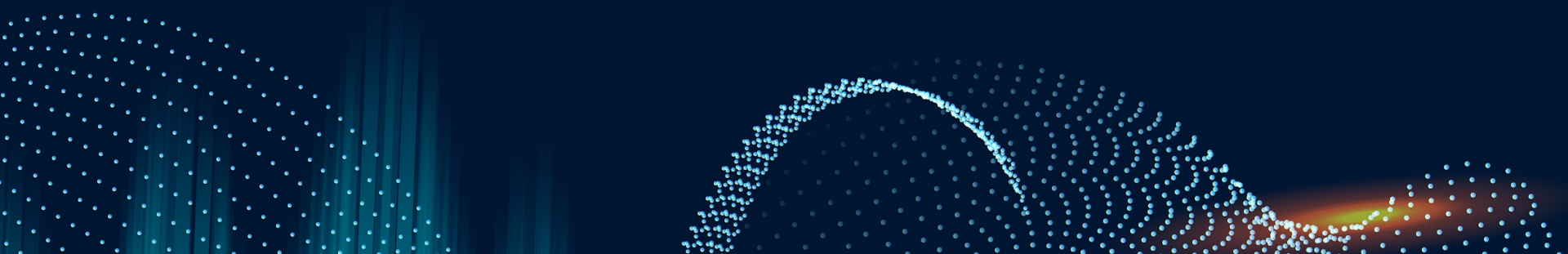
Exemple de vues

Deuxième collection : 1304 enregistrements

Cette vue permet d'avoir tous les enregistrements pour la trompette.

```
pipeline = [{"$match": {"instrument": "trumpet"}}]
view = db.command("create", "vue_instrument_trumpet",
    viewOn="collection", pipeline = pipeline)
```

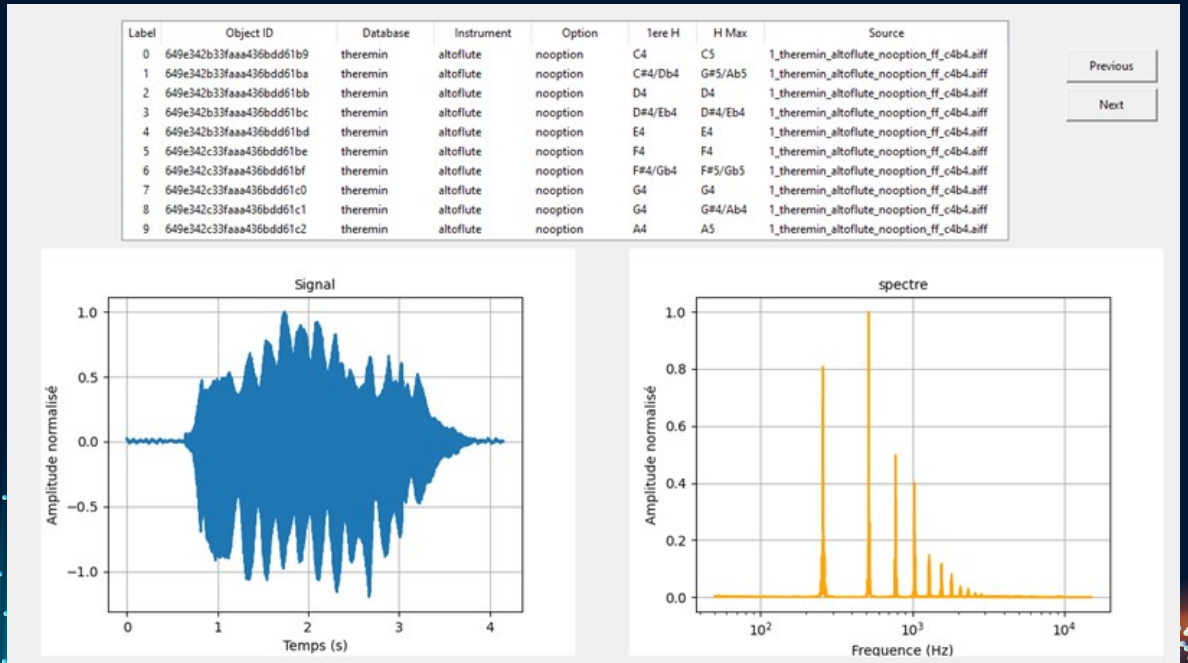
Interaction avec utilisateur – Application projet spectre GUI



Projet Spectre GUI

Deuxième collection : 84 enregistrements

Interface graphique qui permet à l'utilisateur de sélectionner un fichier audio parmi la base de données (84 enregistrements) et d'afficher le signal et le spectre correspondant



Conclusion et perspective

- Proportion d’instrument/option/dynamique où Note max harmonique = Note 1st harmonique et Note max harmonique != Note 1st harmonique
- Exemple : pour Note_max_harmonique = C5, on a 79% de correspondance avec la Note_first_harmonique et 21% de non-correspondance

```
{'Note_max_harmonique': 'C5', 'Note_first_harmonique': 'F3'} : 4
{'Note_max_harmonique': 'C5', 'Note_first_harmonique': 'C5'} : 26
{'Note_max_harmonique': 'C5', 'Note_first_harmonique': 'C4'} : 3
```

- Comportement de l’instrument : par exemple, déterminer les instruments octaviens
- Améliorer le splitter