


June 7, 2018



Universidade Federal de Viçosa - Campus Florestal  
CCF 491 – Tópicos Especiais I  
Trabalho Prático

0.1

Universidade Federal de Viçosa

Aluna (s):	Adriele Dutra Souza	Matrícula (s): 1788
	Juliana Rezende Silveira Baia Alves	1787
	Raissa Polyanna Papini de Melo Souza	2252

## 1 Relatório

A primeira decisão a ser tomada no trabalho prático foi a escolha do *dataset* a ser utilizado. O grupo escolheu o dataset do *site Kaggle* referente a músicas, que basicamente é composto por uma tabela .csv que contém as músicas mais tocadas no *Spotify*. Os dados presentes nesta tabela, foram coletados desde o dia primeiro de janeiro de 2017 até 9 de janeiro de 2018.

A segunda etapa do trabalho consiste na preparação do ambiente para que posteriormente seja possível realizar a análise dos dados. Para isso foram tomadas algumas decisões importantes sobre a formatação e tratamento destes, como por exemplo, a necessidade de aplicar técnicas para eliminar possíveis ruídos que possam interferir nos resultados, e verificar se a estrutura da tabela atende bem aos requisitos.

Foi utilizado o *Jupyter Notebook* como ferramenta principal e a linguagem *Python*, assim como em sala de aula. Após criado o projeto, a primeira coisa feita foi a importação de bibliotecas básicas necessárias e a leitura do arquivo .csv, que não foi difícil pois já se encontrava em uma estrutura fácil de ser lida. A única modificação foi a adição do parâmetro “low\_memory = False” e a retirada do cabeçalho.

```
# Importando bibliotecas necessárias

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Lendo o dataset

musicas = pd.read_csv('data.csv', index_col=False, squeeze=True, low_memory=False);
musicas.rename(columns={'Track Name': 'TrackName'}, inplace=True)
musicas
```

Após a leitura do arquivo, o nome da coluna “Track Name” foi substituído para “Track-Name” para uma melhor manipulação dos dados. Uma vez que o nome das colunas tenham sido padronizados, foi feita uma verificação para analisar se havia alguma célula do arquivo vazia, dentre as 3441197 linhas x 7 colunas apresentadas da tabela. Caso houvesse, o próximo passo seria investigar o motivo da célula estar nula. Mesmo não encontrando ruídos deste tipo, foi implementado um trecho de código para eventuais alterações da tabela. Posteriormente, fizemos uma análise para verificar se havia células com números negativos no *dataset*, mas não obtivemos nenhum resultado.

```
# Verifica se há células nulas

result1 = len(musicas) - pd.isnull(musicas.Position).count()
result2 = len(musicas) - pd.isnull(musicas.TrackName).count()
result3 = len(musicas) - pd.isnull(musicas.Artist).count()
result4 = len(musicas) - pd.isnull(musicas.Streams).count()
result5 = len(musicas) - pd.isnull(musicas.URL).count()
result6 = len(musicas) - pd.isnull(musicas.Date).count()
result7 = len(musicas) - pd.isnull(musicas.Region).count()

resultado = result1 + result2 + result3 + result4 + result5 + result6 + result7
print("Quantidade de celulas nulas: ", resultado)

# Eliminar células com campos vazios

musicas.dropna(axis=0, subset=['Position'], inplace=True)
musicas.dropna(axis=0, subset=['TrackName'], inplace=True)
musicas.dropna(axis=0, subset=['Artist'], inplace=True)
musicas.dropna(axis=0, subset=['Streams'], inplace=True)
musicas.dropna(axis=0, subset=['URL'], inplace=True)
musicas.dropna(axis=0, subset=['Date'], inplace=True)
musicas.dropna(axis=0, subset=['Region'], inplace=True)

musicas

# Verificar se há alguma célula com valor negativo
```

```
musicas[(musicas['Streams'] < 0) | (musicas['Position'] < 0)]
```

Neste ponto, foi possível perceber que haviam caracteres especiais posicionados em lugares indevidos que poderiam atrapalhar na manipulação dos dados e na legibilidade dos dados presentes na tabela. Sendo assim, foi criada uma função para remoção e/ou substituição destes caracteres, removendo, assim, tais ruídos. Além disso, utilizamos uma função para remover espaços em branco do começo e fim dos itens do tipo *String*, para evitar que duas ou mais *strings* iguais sejam diferenciadas entre si.

```
# Eliminar ruídos nos nomes das músicas e artistas

def corrigir (nome):
    nome = nome.replace('#', '').replace('$', 's').replace('*', '')
```

```

    return nome

musicas.Artist = musicas.Artist.apply(corrigir)
musicas.TrackName = musicas.TrackName.apply(corrigir)
musicas

# Tira os espaços em branco do começo e fim da String, para evitar que as mesmas sejam diferentes

musicas['TrackName'] = musicas['TrackName'].str.strip()
musicas['Artist'] = musicas['Artist'].str.strip()
musicas['URL'] = musicas['URL'].str.strip()
musicas['Date'] = musicas['Date'].str.strip()
musicas['Region'] = musicas['Region'].str.strip()

```

Tendo em vista a complexidade de se trabalhar com datas (dia/mês/ano), transformamos esta informação, primeiramente, em dias transcorridos para facilitar na manipulação dos dados e acrescentamos ao *dataframe* original uma coluna com o número(dia) referente à cada data.

```

# Transforma as datas em dias transcorridos para facilitar na manipulação

print ("Primeira Data: {} \n Última Data: {}".format(musicas.Date.min(),musicas.Date.max())) # Verifica o formato da data
Dates = pd.to_datetime(musicas.Date) # Verifica o formato da data
Days = Dates.sub(Dates[0], axis = 0) # Subtrai resultados redundantes
Days = Days / np.timedelta64(1, 'D') # converte para Float
print ("Primeiro Dia: {} \n Último Dia: {}".format(Days.min(), Days.max())) # check converted first
musicas['Days'] = Days # Adiciona a nova coluna com Float's ao dataframe

```

Ainda em relação às informações sobre datas, outra alternativa que utilizamos também para facilitar a manipulação dos dados foi utilizar apenas o mês e o ano. Fazendo isso é possível extrair estatísticas sobre um período de tempo mais específico.

```

#Classifica a data apenas em ano e mês

musicas['Year'] = musicas.Date.str[:4]
musicas['Month'] = musicas.Date.str[5:7]

```

Além das alterações e informações apresentadas anteriormente, foi necessário também, pesquisar sobre as regiões presentes no *dataset*, uma vez que as mesmas são identificadas apenas por sua respectiva sigla, o que pode causar um pouco de confusão no momento de analisar as informações extraídas do conjunto de dados referentes ao atributo "Region". Segue abaixo a tabela contendo a sigla e seu respectivo nome.

Sigla	Região
ar	Argentina
at	Austria
au	Australia
be	Belgium
bo	Bolivia

Sigla	Região
br	Brazil
ca	Canada
ch	Switzerland
cl	Chile
co	Columbia
cr	CostaRica
cz	CzechRepublic
de	Germany
dk	Denmark
do	DominicanRepublic
ec	Ecuador
ee	Estonia
es	Spain
fi	Finland
fr	France
gb	UnitedKingdom
global	World
gr	Greece
gt	Guatemala
hk	HongKong
hn	Honduras
hu	Hungary
id	Indonesia
ie	Ireland
is	Iceland
it	Italy
jp	Japan
lt	Lithuania
lu	Luxemborg
lv	Latvia
mx	Mexico
my	Malaysia
nl	Netherlands
no	Norway
nz	NewZealand
pa	Panama
pe	Peru
ph	Philippines
pl	Poland
pt	Portugal
py	Paraguay
se	Sweden
sg	Singapore
sk	Slovakia
sv	ElSalvador
tr	Turkey

Sigla	Região
tw	Taiwan
us	USA
uy	Uruguay

De acordo com a descrição deste trabalho prático, a terceira etapa consiste em explorar e extrair informações, a fim de se gerar estatísticas, gráficos e/ou tabelas para um melhor entendimento dos dados. Além de realizar análise da correlação entre todos os atributos e objetos possíveis.

Primeiramente, realizamos cálculos estatísticos com base no atributo "Streams" utilizando a função `describe()` que é proveniente da biblioteca `pandas`.

```
python # Cálculos estatísticos com base na coluna "Streams" utilizando a função describe()
musicas['Streams'].describe()
```

Prosseguindo a análise, foi possível perceber, através do comando `value_counts()`, aplicado à coluna 'Track Name', que das 3441197 linhas presentes na tabela, somente 18597 músicas são distintas, o que pode soar um pouco estranho. Este resultado pode se dar pelo fato de que muitos artistas possuem músicas com nomes iguais, interferindo, assim, no retorno da função utilizada.

```
# Verificar a quantidade de músicas distintas no dataset
```

```
Nomes = musicas['TrackName']
totalNomes = len(Nomes.value_counts())
print( "O total de nome de musicas distintas é de : ", totalNomes)
print("E o tamanho do nosso dataset é de ", len(musicas), "linhas (musicas)")
```

Com o intuito de analisar o comportamento dos streams, os dados foram agrupados por região e, assim, foi possível visualizar melhor o seu comportamento. Devido a isso, agrupamos por região os dados para verificarmos se alguma região estava se apresentando de forma errada na tabela, caso o GPS tenha coletado algo erroneamente ou o usuário tenha digitado errado. Agrupando, é possível verificar se há alguma região errada, com nome diferente, ou inexistente.

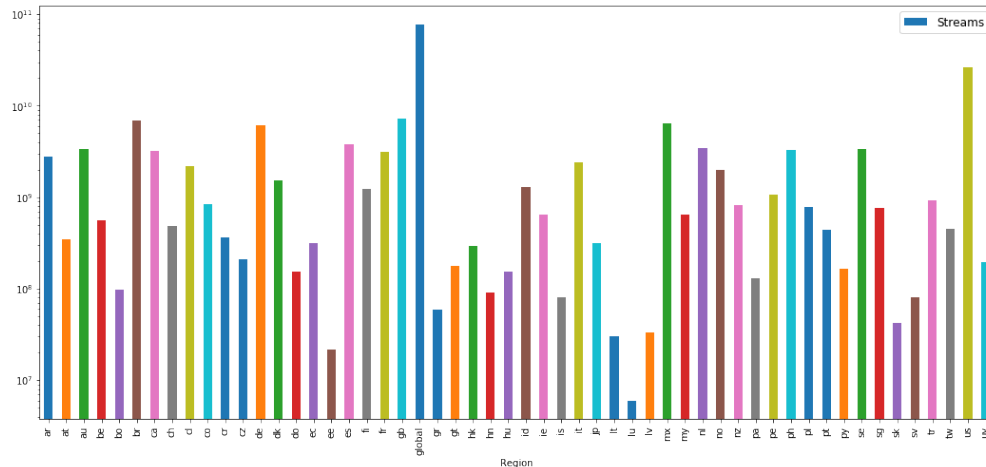
```
# Quantidade de Streams por região
```

```
paises = musicas.groupby('Region')
paises_sum = paises.sum()
paises_sum['Streams'].plot(kind='bar')
plt.yscale('log')
plt.rcParams['figure.figsize'] = (18,8)
plt.legend(loc='upper right', prop={'size':12}, fontsize=1)
plt.show()
```

Seguindo a mesma linha de raciocínio utilizada para ver a quantidade de *streams* por região, analisamos também a quantidade de *streams* agrupando estes por dia e por mês. Possibilitando uma melhor análise do atributo "Streams" em amostras menores.

```
#Quantidade de Streams por Dia/Data
```

```
dias = musicas.groupby(['Days', 'Date'])
dias_sum = dias.sum()
```



dias\_sum

### #Quantidade de Streams por mês

```
meses = musicas.groupby(['Year', 'Month'])
meses_sum = meses.sum()
meses_sum
```

Com base nos agrupamentos realizados acima, pudemos gerar um gráfico apresentando um *ranking* dos cinco meses com mais *streams*.

## #Ranking dos 5 meses com mais Streams

```
meses_sum['Streams'].sort_values(ascending=False).head().plot(kind='bar')
plt.rcParams['figure.figsize'] = (4,2)
plt.legend(loc='upper right', prop={'size':10}, fontsize=5)
plt.show()
```

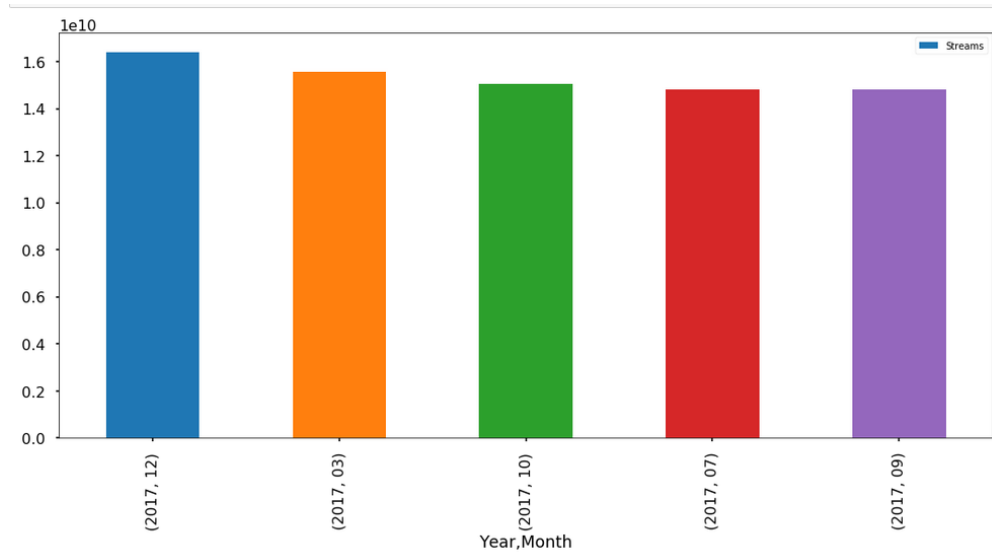
Associando a quantidade de *streams* com outros atributos do *dataset*, é possível gerar outras estatísticas para análise. Após manipularmos estas informações com base na data e região, fizemos também a correlação entre os atributos *Streams* e *Artist*, e entre os atributos *Streams* e *TrackName*. Limitamos a visualização para apenas vinte e cinco itens e apresentamos então um *ranking* dos 25 artistas mais acessados e das 25 músicas mais acessadas, respectivamente.

### #Ranking dos 25 artistas mais acessados

```
artistas = musicas.groupby('Artist')
artistas_soma = artistas.Streams.sum()
```

```
#Cria um novo dataframe para manipulação
```

```
contStreams=pd.DataFrame(artistas_soma)
contStreamsM = contStreams.Streams.sort_values(ascending=False)
```



Streams por Mês

```
contStreamsM = contStreamsM[:25]

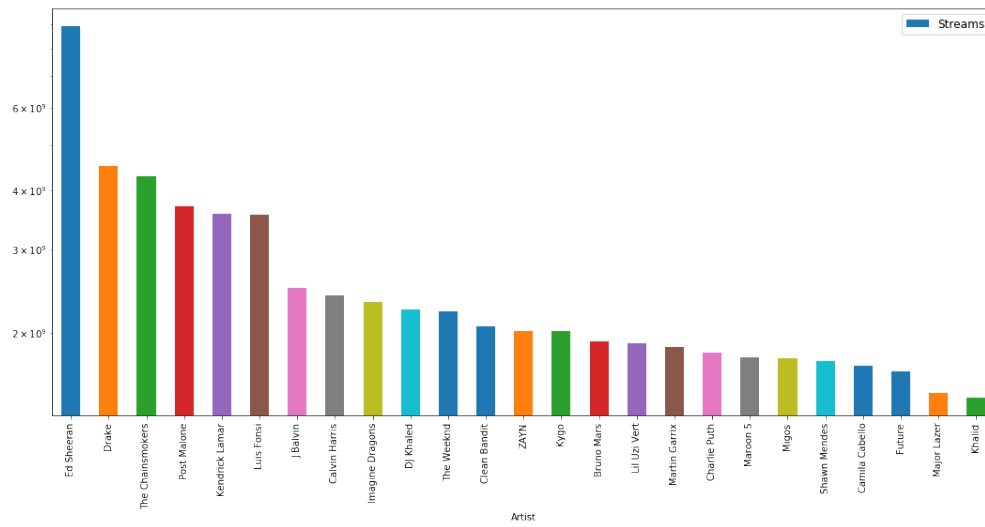
contStreamsM.plot(kind='bar')
plt.rcParams['figure.figsize'] = (18,8)
plt.legend(loc='upper right', prop={'size':12}, fontsize=1)
plt.yscale('log')
plt.show()

#Ranking das 25 músicas mais acessadas
musicas_mais = musicas.groupby('TrackName')
musicas_mais_soma = musicas_mais.Streams.sum()

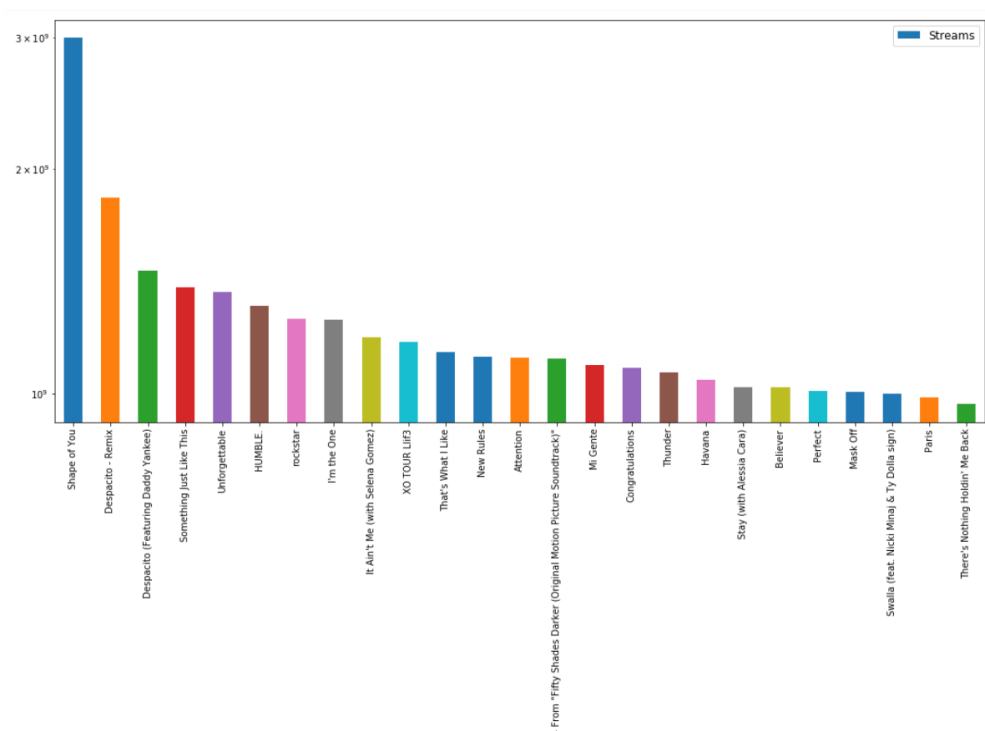
#Cria um novo dataframe para manipulação
contMusics=pd.DataFrame(musicas_mais_soma)
contMusicsM = contMusics.Streams.sort_values(ascending=False)
contMusicsM = contMusicsM[:25]

contMusicsM.plot(kind='bar')
plt.rcParams['figure.figsize'] = (18,8)
plt.legend(loc='upper right', prop={'size':12}, fontsize=1)
plt.yscale('log')
plt.show()
```

Em seguida foi feito um gráfico de dispersão no qual o eixo x representa as datas e o eixo y representa as Streams. Este gráfico foi feito pensando que, durante um ano, as visualizações de músicas podem ser mais intensas ou não, e esta análise é interessante para, por exemplo, saber quando colocar mais anúncios em um aplicativo por saber que o fluxo de streams é mais intenso naquela época do ano. Ou é também para saber quando um artista ou vários artistas lançam mais



Ranking dos 25 artistas mais acessados



Ranking das 25 músicas mais acessados



músicas e outras análises podem ser tiradas, a partir deste gráfico, por especialistas. O gráfico pega todas as datas presentes do nosso dataset, porém foi feita uma filtragem dos xticklabels, para melhorar a visualização do gráfico, pois se não houvesse tal seleção dos xticklabels, não seria possível a leitura das datas, tornando o gráfico ilegível.

```
fig = gcf()
ax = fig.gca()
plt.style.use('seaborn-poster')

y = musicas['Streams']
x = musicas['Date']

plt.title('Streams por Data', fontsize='14')
plt.xlabel('Data', fontsize='14')
plt.ylabel('Streams', fontsize='14')

#plt.rcParams['figure.figsize'] = (100,18)
#plt.plot(x, y, 'o', color='blue');
plt.scatter(x, y, color='blue', s=50, alpha=.5);

xmarks=['2017-01-01', '2017-03-18', '2017-08-30', '2018-01-09'] # 5 registros
plt.xticks(xmarks)

labels = [item.get_text() for item in ax.get_xticklabels()]
labels[0] = '2017-01-01'
labels[1] = '2017-03-18'
labels[2] = '2017-08-30'
labels[3] = '2018-01-09'
ax.set_xticklabels(labels)

plt.savefig('StreamsPorData.png')
```

Foi feito também um outro gráfico de dispersão, no qual o eixo x representa o nome do cantor e o eixo y representa a track name, que seria o nome da música, sendo assim, gera uma relação que representa a música de cada cantor. Além disso foi representada uma outra camada de cor, através de um colorbar, que representa as Streams, dessa forma, mostra para nós uma análise das músicas mais tocadas de cada cantor. Essa análise é muito interessante pois se rodada com o dataset inteiro mostraria todas as músicas mais tocadas de todos os cantores.

```
#Ranking dos 50 musicas mais acessadas, com seus artistas
artistas_mus = musicas.groupby(['Artist', 'TrackName'])
artistas_mus = artistas_mus.Streams.sum()
artistas_mus
#Cria um novo dataframe para manipulação
artistas_musf=pd.DataFrame(artistas_mus)
artistas_musf = artistas_musf.Streams.sort_values(ascending=False)
```

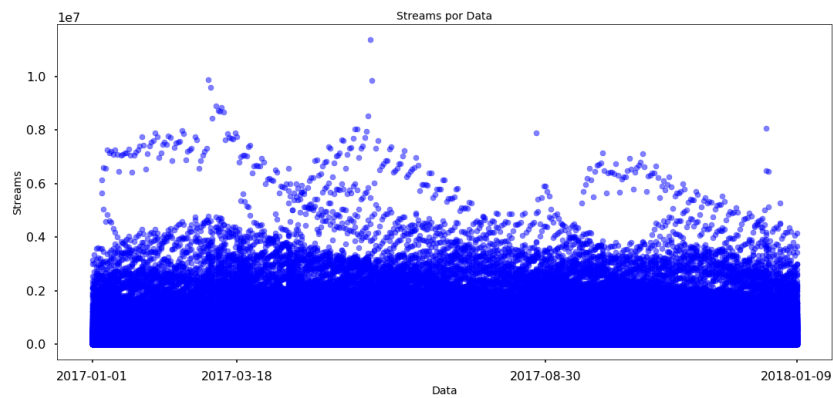


Gráfico de Dispersão - Streams por Data

```

artistas_musf = artistas_musf[:50]
art = artistas_musf.reset_index()
art

fig = gcf()
ax = fig.gca()
plt.style.use('seaborn-poster')

X = art['Artist']
Y = art['TrackName']
C = art['Streams']

plt.rcParams['figure.figsize'] = (100,50)

# cmap = mpl.colors.ListedColormap([ 'tab:red', 'tab:orange', 'tab:red'])

s = ax.scatter(X,Y,c=C,lw=0.3, s=500, alpha=0.5, cmap='autumn') #plasma, hsv 'viridis', 'plasma

plt.title('Gráfico de artistas por Track name em função das streams.', fontsize='22')
plt.ylabel('Track Name', fontsize='14')
plt.xlabel('Artist', fontsize='14')

# norm = mpl.colors.Normalize(vmin=1, vmax=20)
cb = plt.colorbar(s)

cb.set_label('Streams', fontsize='14')

# plt.savefig('ArtistasPorStreamsCores.png')

```

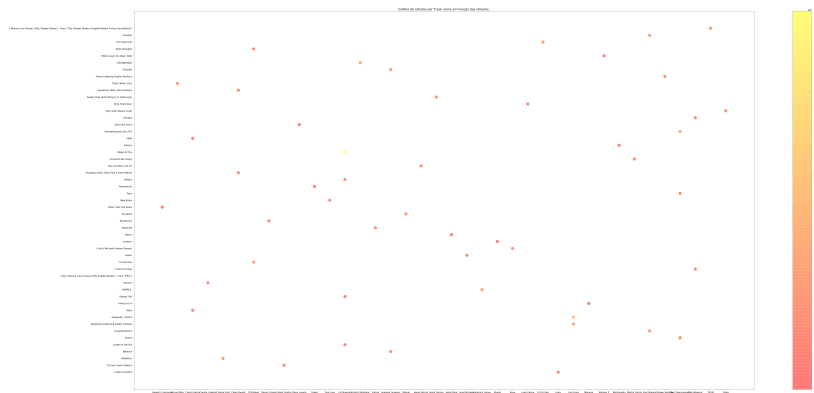


Gráfico de Dispersão - Artistas por Streams