Recueil de données

Naila BOUTERFA, Thomas DINH, Wendy LEFEVRE

07 janvier, 2019

Introduction

Dans le cadre de notre projet de recueil de données nous allons nous intéresser à des données tirées d'API de sites de streaming musical tel que Deezer et Spotify. Les données dans un premier temps nous mettent face à une problématique data munging et nous allons consacrer la première partie de ce rapport à une importation appropriée de ces données diverses, en plus de ça nous allons supprimer certaines variables, en modifier d'autres et fusionner des tables pour faciliter les étapes d'analyse et d'exploitation des données qui suivent. Globalement, nous avons deux types de variables, des variables liées à la nature du son et des variables liées aux artistes (à quel point ils sont prolifiques, leur succès...)

Importation de données

Dans deezer nous avons :

```
playlist=read.table(file="C:/Users/wendy/Documents/M2_RECCUEIL/Projet recueil/playlists2.txt",fill=TRUE
Tracks = read_delim("C:/Users/wendy/Documents/M2_RECCUEIL/Projet recueil/Tracks.csv",delim=',')
Artists = read_delim("C:/Users/wendy/Documents/M2_RECCUEIL/Projet recueil/Artists.csv",delim=',')
tracks_proj = read_delim("C:/Users/wendy/Documents/M2_RECCUEIL/Projet recueil/tracks_proj.csv",delim =
```

Dans spotify nous avons :

```
Artist_details_spot = read_delim("C:/Users/wendy/Documents/M2_RECCUEIL/Projet recueil/artist_details_sp
track_details_spot = read.table(file = "C:/Users/wendy/Documents/M2_RECCUEIL/Projet recueil/track_detai
```

Fichiers Deezer :

Traitement des données

Nous allons nous concentrer dans un premier temps sur les fichiers provenant de l'API Deezer :

Nous sélectionnons les variables significatives de Tracks et en purifions certaines qui contiennent les identifiants d'artistes ou de titres afin de pouvoir obtenir une variable numérique que nous exploiterons pour mettre en relation nos différentes tables.

```
new_id = c()
for (i in 1:4237){
  test = str_extract_all(string = Tracks$artist[i], boundary("word"))[[1]]
  n = which(str_detect(test, "u'id"))
  new_id[i] = test[n+1]
}
new_id[4238] = 1612447
for (i in 4239:nrow(Tracks)){
```

```
  test = str_extract_all(string = Tracks$artist[i], boundary("word"))[[1]]
  n = which(str_detect(test, "u'id"))
  new_id[i] = test[n+1]
}
Tracks = Tracks %>% mutate(id_artist = new_id)

Tracks=Tracks%>%select(c("id","id_artist","title","isrc","available_countries","bpm","disk_number","dura
```

Sélection de variables dans $track_proj$: nous filtrons encore les variables d'intéret qui sont interprétables à l'échelle de notre analyse.

```
tracks_proj=tracks_proj%>%select(c("id","bpm","duration","gain","release_date","title","album_title","a
```

De même nous procédons à la modification de la variable id dans $tracks_proj$: on veut avoir un identifiant manipulable donc une variable numérique.

```
id = tracks_proj$id
for (i in 1:length(id)){
  id[i] = substr(id[i], start = 7, stop = str_count(id[i]))
}
id = as.numeric(id)
tracks_proj$id = id
```

Ensuite nous fusionnons les fichiers $Tracks$ afin d'obtenir une table plus complète :

```
Tracks=merge(x = Tracks,y = tracks_proj,by=c("id","bpm","duration","gain","release_date","title"))
rm(tracks_proj)
```

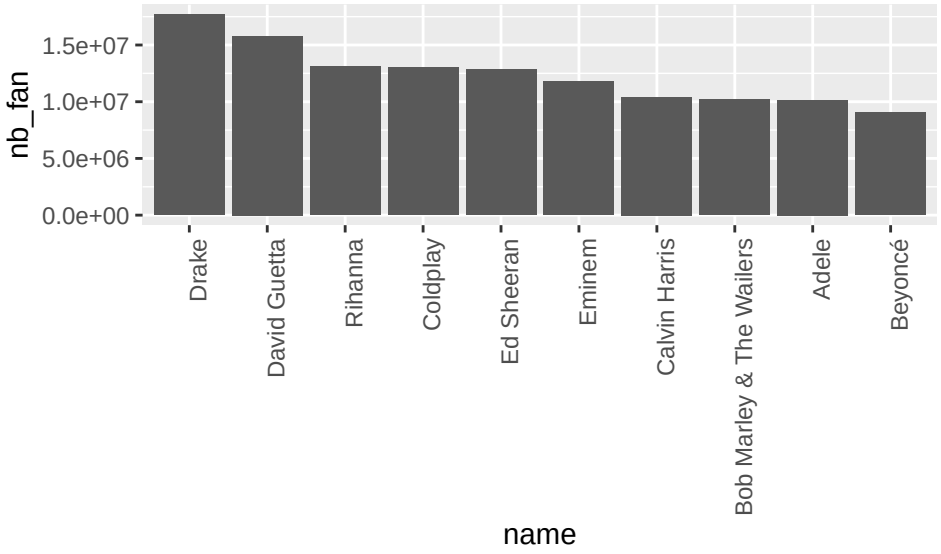
De même pour le fichier $Artists$, nous purifions le fichier des données inutiles et les données manquantes.

```
Artists=Artists[-which(!is.na(Artists$error)),]
Artists=Artists%>%select(c("id","name","nb_album","nb_fan","radio"))
```

Visualisation de données Deezer :

Nous pouvons faire quelques visualisations d'intérêt sur nos fichiers Deezer. Ci-dessous nous observons le top 10 des artistes ayant le plus de fans, classé du plus grand nombre au plus petit. En tête de file : Drake, David Guetta et Rihanna.

```
top_fan = Artists %>% arrange(desc(nb_fan)) %>%
  head(10)

top_fan %>% arrange(desc(nb_fan)) %>%
  mutate(name=factor(name,levels=name)) %>%
  ggplot()+
  geom_bar(aes(x=name,y=nb_fan),stat = 'identity')+
  scale_x_discrete("name")+
  scale_y_continuous("nb_fan") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
rm(top_fan)
```

Ici nous pouvons chercher à retrouver les chansons les plus longues et les plus courtes de notre base de données, avant cela nous convertissons nos données temps en seconde en quelque chose de plus lisible sous forme (minute, secondes)

```
duration = Tracks$duration
for (i in 1:length(duration)){
  Min_sec = seconds_to_period(duration[i])
  duration[i] = sprintf('%02d:%02d',minute(Min_sec), second(Min_sec))
}
knitr::kable(cbind(durée_max = max(duration),durée_min = min(duration)), caption = "durée maximale et mi
```

Table 1: durée maximale et minimale de chanson

durée_max	durée_min
15:41	00:38

Nous pouvons également identifier les titres ayant la plus grande et plus faible mesure *bpm*, cela donne :

```
max_bpm=max(Tracks$bpm)
a=Tracks[which(Tracks$bpm==max_bpm),c("title","artist_name","bpm")]
#min_bpm= min(Tracks$bpm)
min_bpm=min(Tracks[which(Tracks[,"bpm"]!=0),"bpm"])
b=Tracks[which(Tracks$bpm==min_bpm),c("title","artist_name","bpm")]

knitr::kable( rbind(bpm_max = a, bpm_min = b), caption = "Musique de battements par minutes minimal et m
```

Table 2: Musique de battements par minutes minimal et maximal

	title	artist_name	bpm
bpm_max	Mister Sandman (Featuring Al Roberts)	The Four Aces	219
bpm_min	Générique (BOF “Ascenseur pour l’échafaud”)	Miles Davis	69

De même si l’on s’intéresse cette fois aux titres de plus haut rang on obtient le classement suivant :

```
top_rank = Tracks %>% arrange(desc(rank)) %>% head(5)
knitr::kable(top_rank[,c("rank","title","artist_name")])
```

rank	title	artist_name
1000000	Shape of You	Ed Sheeran
998101	I Will Survive (Single Version)	Gloria Gaynor
997411	Look At Me!	Xxxtentacion
995244	Perfect	Ed Sheeran
994926	Something Just Like This	The Chainsmokers

Fichiers Spotify

Traitement des données

Nous allons maintenant procéder de la même manière avec les données de l’API Spotify.

Nous sélectionnons les variables d’intérêt et purifions les variables contenant des identifiants

```
Artist_details_spot=Artist_details_spot%>%select(c("deezer_id","genre","name","popularity"))
```

```
deezer_id = track_details_spot$deezer_id
for (i in 1:length(deezer_id)){
  deezer_id[i] = substr(deezer_id[i], start = 7, stop = str_count(deezer_id[i]))
}
deezer_id = as.numeric(deezer_id)
track_details_spot$deezer_id = deezer_id

track_details_spot=track_details_spot%>%select(c("deezer_id","acousticness","danceability","duration_ms
```

Visualisation de données Spotify :

Nous avons un certain nombre de variables relatives au son dans ces bases de données. Nous pouvons les voir dans le tableau ci dessous :

deezer_id	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	
100001884	0.10700	0.792	195637	0.743	0.0e+00	0.1830	-2.806	0.0851	1
100004586	0.08200	0.613	239920	0.765	1.3e-06	0.0920	-6.582	0.0441	1
100004588	0.00784	0.532	238200	0.599	0.0e+00	0.1400	-6.543	0.0333	1
100004590	0.05910	0.748	235493	0.788	0.0e+00	0.0863	-7.055	0.0334	1

deezer_id	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	
100004592	0.28400	0.501	203453	0.704	0.0e+00	0.1550	-5.640	0.0326	
100004594	0.03200	0.618	219493	0.837	0.0e+00	0.0475	-6.142	0.0515	1

De même dans les tables issues de l'API Spotify, nous avons une information supplémentaire au sujet des artistes qui est le genre musical dans lequel ils s'inscrivent. Cette variable *genre* nous sera très utile par la suite.

deezer_id	genre	name	popularity
7912872	tropical house	Petit Biscuit	71
1188	pop	Maroon 5	90
14803	NA	Toploader	63
7218	latin	Gloria Trevi	71
5068032	pop	Felix Jaehn	76
535490	hardstyle	Ran-D	59

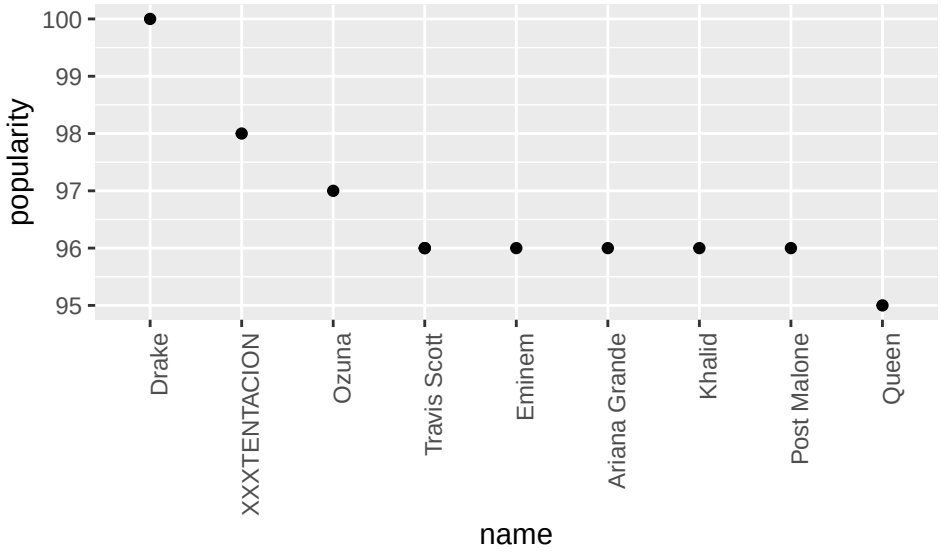
Parmi les genres musicaux les plus représentés dans notre base de données, nous avons ce qui suit :

```
genres=table(Artist_details_spot$genre)
names(which(genres>100))
```

```
##  [1] "chanson"        "dance pop"      "folk-pop"       "french hip hop"
##  [5] "latin"          "modern rock"    "mpb"            "pop"
##  [9] "rap"            "rock"           "tropical house"
```

De même nous pouvons observer le top 10 des artistes les plus populaires dans le graphique ci dessous. Avec toujours en tête Drake, suivis de XXXtentacion et d'Ozuna.

```
top_popularity = Artist_details_spot %>% arrange(desc(popularity)) %>%
  head(10)%>% mutate(name=factor(name,levels = unique(name)))

top_popularity  %>%
  ggplot()+  geom_point(aes(x=name,y=popularity))+
  scale_x_discrete("name")+
  scale_y_continuous("popularity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
rm(top_popularity)
```

Etude de Corrélations

Nous constatons à travers ce tableau de t-valeurs de tests de corrélation que nos variables sonores sont très corrélées, ce qui est plutot cohérent.

```
library(psych)
Cor=corr.test(track_details_spot[,2:12],adjust="none")
Tab=lowerMat(Cor$p,digits=10)
```

Fusion de tableaux et analyse approfondie

Nous allons dans ce qui suit créer une grande table de données en faisant des jointures par ID ce qui nous permettera d'avoir des données plus complètes pour la suite de notre analyse :

```
Tracks2=merge(x = track_details_spot,y = Tracks,by.x = "deezer_id",by.y = "id")
Tracks2=merge(x = Tracks2,y = Artist_details_spot,by.x = "id_artist",by.y = "deezer_id")
```

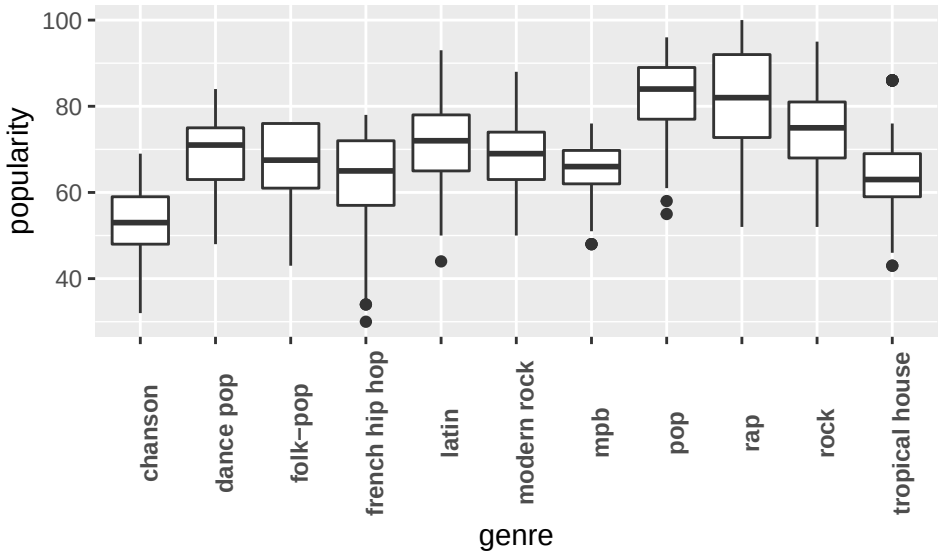
Boxplots sur les genres musicaux

Nous allons dans ce qui suit nous concentrer sur les genres musicaux les plus représentés que l'on aura cité précedemment.

```
genre = table(Artist_details_spot$genre)
genre_important = names(genre[genre > 100])
```

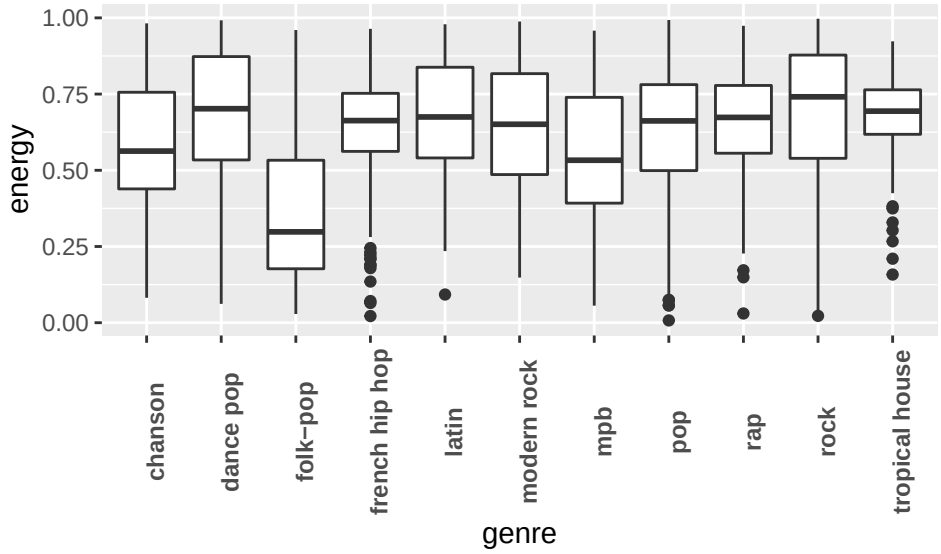
Nous avons ainsi les boxplots représentatifs de la popularité pour chaque genre musical de notre sélection. Les genres les plus populaires semblent être la pop et le rap.

```
Tracks2 %>% filter(genre %in% genre_important) %>%
  ggplot()+
  geom_boxplot(mapping = aes(x = genre, y = popularity), na.rm = TRUE) + #boxplot
  theme(axis.text.x = element_text(face="bold", size=9, angle=90))
```



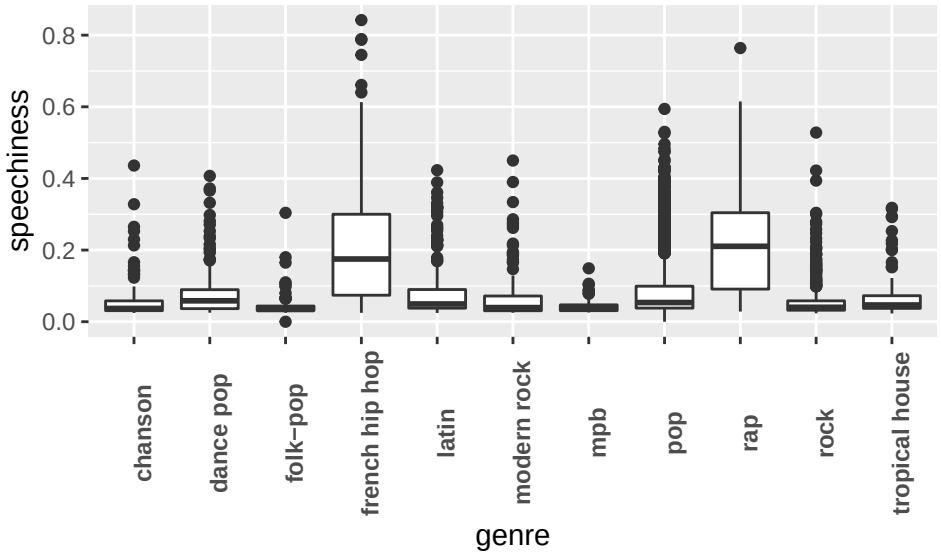
Ci dessous nous pouvons observer la répartition des genres musicaux les plus énergiques, la folk-pop semble être le genre musical le moins énergique.

```
Tracks2 %>% filter(genre %in% genre_important) %>%
  ggplot()+
  geom_boxplot(mapping = aes(x = genre, y = energy), na.rm = TRUE) + #boxplot
  theme(axis.text.x = element_text(face="bold", size=9, angle=90))
```



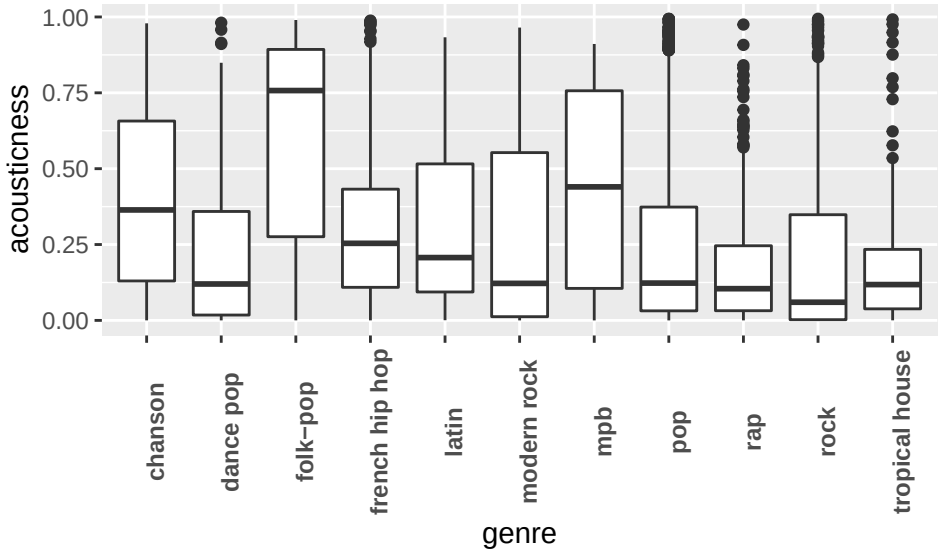
Pour ce qui est des paroles, le rap et le french hip-hop semblent être les genres musicaux les plus fournis en texte.

```
Tracks2 %>% filter(genre %in% genre_important) %>%
  ggplot()+
  geom_boxplot(mapping = aes(x = genre, y = speechiness), na.rm = TRUE) + #boxplot
  theme(axis.text.x = element_text(face="bold", size=9, angle=90))
```



Pour ce qui est du caractère acoustique de la musique on retrouve naturellement la folk-pop suivi du style mbp (Musique populaire brésilienne).

```
Tracks2 %>% filter(genre %in% genre_important) %>%
  ggplot()+
  geom_boxplot(mapping = aes(x = genre, y = acousticness), na.rm = TRUE) + #boxplot
  theme(axis.text.x = element_text(face="bold", size=9, angle=90))
```



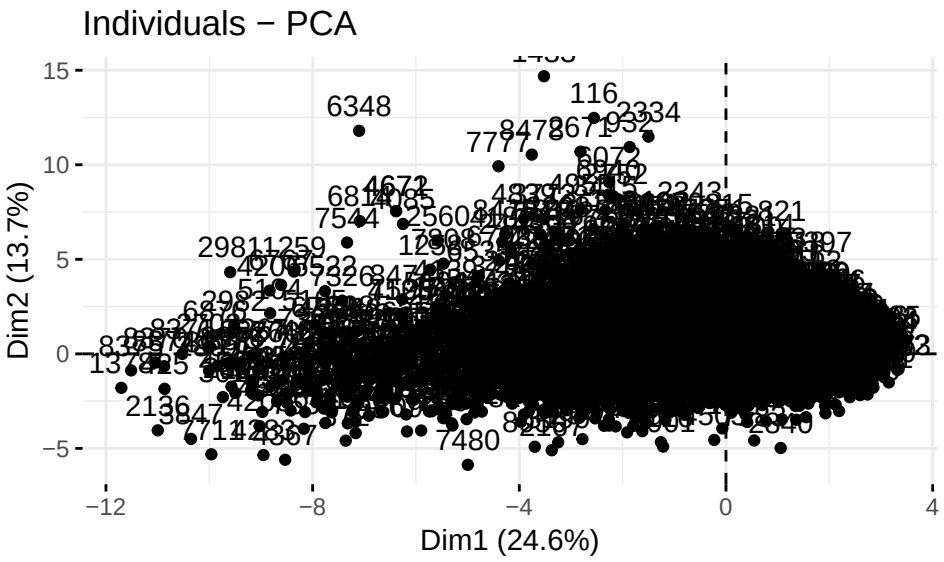
Analyse de composante principale :

Nous procédons à une Analyse de composantes principales sur les variables quantitatives de notre grande base de données. Nous pouvons observer les pourcentages de variances associées à chaque composante, les graphique en 2D de la projection des individus sur ses composantes ainsi qu'un corrplot.

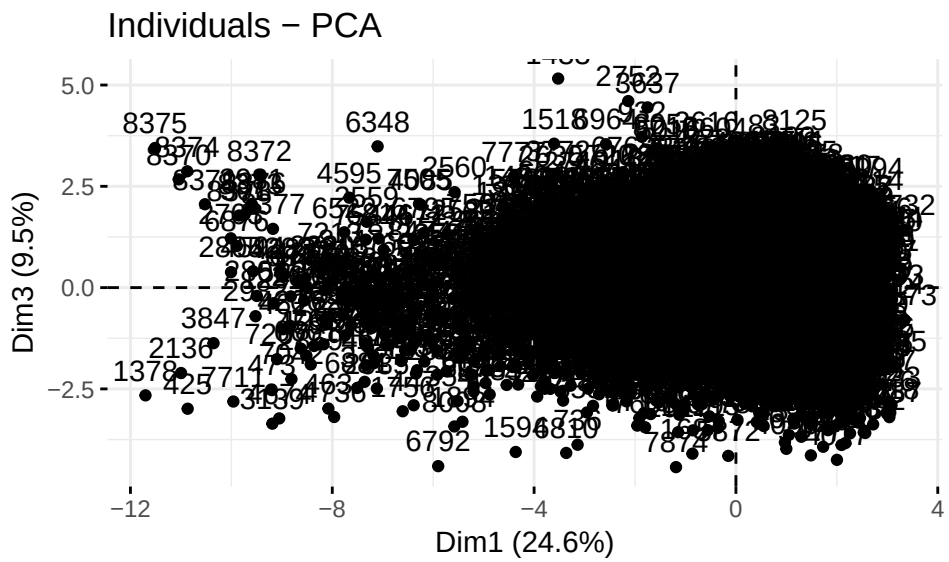
```
ACP_tracks=PCA(Tracks2[,3:16], scale.unit = TRUE, graph = FALSE)

get_eigenvalue(ACP_tracks)[1:5,]
```

```
##       eigenvalue variance.percent cumulative.variance.percent
## Dim.1  3.4463345        24.616675                    24.61667
## Dim.2  1.9217150        13.726536                    38.34321
## Dim.3  1.3348785         9.534846                    47.87806
## Dim.4  1.1265593         8.046852                    55.92491
## Dim.5  0.9888607         7.063291                    62.98820
```

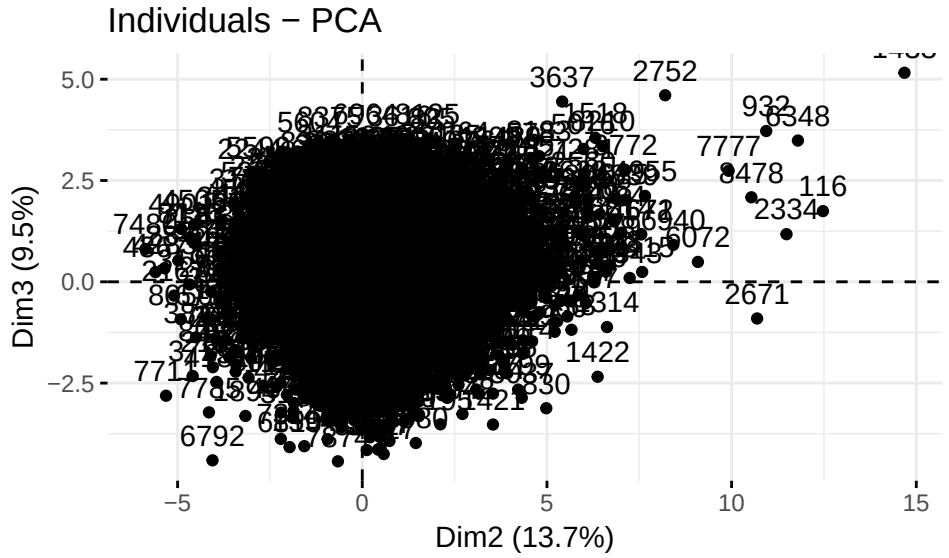
```
fviz_pca_ind(ACP_tracks)
```



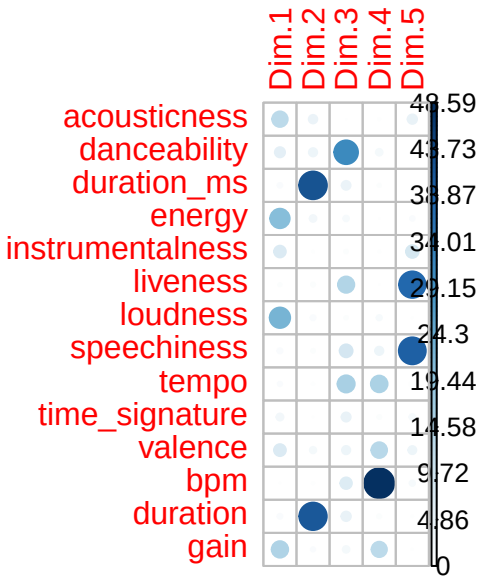

```
fviz_pca_ind(ACP_tracks,c(1,3))
```




```
fviz_pca_ind(ACP_tracks,c(2,3))
```




```
corrplot(ACP_tracks$var$contrib, is.corr=FALSE)
```



Classification des genres par la méthode des plus proches voisins

Nous allons utiliser les composantes princpales résultant de notre ACP pour effectuer une classification en se basant sur ses variables pour l'apprentissage et afin de prédire le genre musical associé à tout individu.

La prédiction donne ce qui suit :

```
library(class)

genre = table(Tracks2$genre)
genre_important = names(genre[genre > 200])

Tracks2class=Tracks2%>%filter(genre%in%genre_important)%>%select(c("acousticness","danceability","durati

ACP_var=as.data.frame(ACP_tracks$ind)[,1:4]
PCA_classif=ACP_var%>%mutate(genre=Tracks2$genre)%>%filter(genre%in%genre_important)%>%mutate(genre=as.

Classif_genre=knn(PCA_classif[1:4000,1:4],PCA_classif[4001:4115,1:4],cl=PCA_classif[1:4000,5],k=17)

Tracks3class = Tracks2class%>%mutate(prediction = c(Tracks2class[1:4000,]$genre,Classif_genre), genre_nu

head(Tracks3class[4001:4115,17:19])
```

```
##               genre prediction genre_num
## 4001            pop          5         3
## 4002           rock          3         5
## 4003           rock          6         5
## 4004           rock          5         5
## 4005           rock          5         5
## 4006 french hip hop          3         1
```

Avec un taux d'erreur de :

```
nb_error=sum(Tracks3class[4001:4115,18]!=Tracks3class[4001:4115,19])
t=(nb_error/115)*100

t
```

```
## [1] 43.47826
```

Cartes

Nous allons maintenant nous intéresser à l'emplacement géographique des différentes musiques de nos bases de données : Pour cela, on extrait de la variable ISCR les deux premiers caractères représentatifs du pays (norme ISO) d'où provient chaque musique et on stocke le pays dans une nouvelle variable.

```
Tracks = Tracks %>% mutate(Pays = substr(Tracks$isrc, start = 1, stop = 2))
# paysFR <- read.table("C:/Users/Naila/Desktop/Recueil_de_données/paysFR.txt", sep=":", header = F)
paysFR <- read.table("C:/Users/wendy/Documents/M2_RECCUEIL/Projet recueil/paysFR.txt", sep=":", header =
PaysFR = paysFR[which(paysFR$V1 %in% Tracks$Pays),]
```

Nous avons également importé d'un fichier extérieur sur le site https://www.maxmind.com/fr/free-world-cities-database, les coordonnées des plus grandes villes du monde (latitude et longitude). Le code ISCR ne donne que le pays d'enregistrement de la musique donc nous avons fait la moyenne de ces coordonnées des villes par pays. Nous pouvons donc placer sur la carte chaque pays qui sont contenus dans notre base de données. Or ce qui nous interesse c'est de voir en un coup d'oeil le nombre de musique enregistré par pays.

```
# worldcitiespop = read.table("C:/Users/Naila/Desktop/Recueil_de_données/worldcitiespop.txt", sep=",",
worldcitiespop = read.table("C:/Users/wendy/Documents/M2_RECCUEIL/Projet recueil/worldcitiespop.txt", se


worldcitiespop$Country = str_to_upper(worldcitiespop$Country)
worldcitiespop = worldcitiespop %>%
  filter(Country %in% PaysFR$V1) %>%
  filter(Longitude != "")

worldcitiespop$Latitude = as.numeric(worldcitiespop$Latitude)
worldcitiespop$Longitude = as.numeric(worldcitiespop$Longitude)

coordonnees = worldcitiespop %>%
  group_by(Country) %>%
  summarize(lat = mean(Latitude, na.rm = TRUE), long = mean(Longitude, na.rm = TRUE))
```

```
Tracks = merge(x = Tracks, y = coordonnees, by.x = "Pays", by.y = "Country")

T_pays = Tracks %>% group_by(Pays) %>%
  summarize(table(Pays))

Tracks = merge(x = Tracks, y = T_pays, by = "Pays")
```

Nous avons tracé la carte du monde avec les provenances des différentes musiques. Plus le point est foncé, plus le nombre de musiques enregistrées est grand. On observe principalement une concentration de points en Europe (surtout en France avec plus de 3000 musiques) et en Amérique (plus de 5000 musiques).

```r
my_sf = st_as_sf(x = Tracks, coords = c("long", "lat"), crs = 4326)

leaflet(data = my_sf ) %>%
  addTiles() %>%
  addCircleMarkers(radius =~ sqrt(table(Pays)*0.01))
```

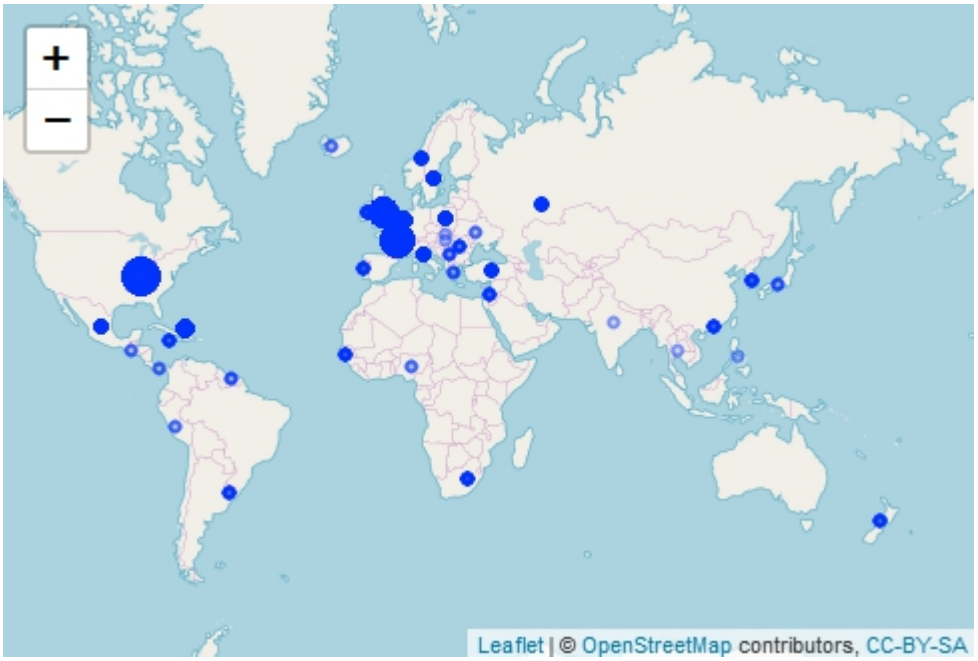


Figure 1: Carte représentant les pays où les musiques sont les plus enregistrées selon la taille et l'opacité à l'aide des librairies Leaflet et sf

Recommandations sur playlist

Nous nous intéressons maintenant à la construction d'un algorithme permettant de recommander des musiques à partir d'une playlist déjà définie. Pour cela nous allons utiliser les variables du tableau fusionné. Nous traitons tout d'abord les données du fichier *playlist* pour n'extraire que l'ID de la musique puis nous l'associons au tableau fusionné. Ensuite lorsque nous avons notre playlist, nous faisons la moyenne des variables quantitatives et nous regardons la musique qui s'en rapproche le plus.

```r
numero = function(playlist){
  playlist = playlist[which(playlist != "")]
  for(i in seq(1,length(playlist),2)){
    playlist[i] = substr(playlist[i], start = 7, stop = str_count(playlist[i]))
  }
  for(i in seq(2,length(playlist),2)){
    playlist[i] = substr(playlist[i], start = 8, stop = str_count(playlist[i]))
  }
  playlist = as.numeric(playlist)
  return(playlist)
}
```

```
recommendation = function(playlist){
  playlist = numero(playlist)
  playlist_track_details = Tracks2 %>% filter(deezer_id %in% playlist[seq(1,length(playlist),2)])
  moyenne_variables = apply(X = playlist_track_details[,c("acousticness","danceability","energy","liven
  musique_recommandee = rep(NA,nrow(Tracks2))
  Tracks3 = Tracks2[-c(which(Tracks2$deezer_id %in% playlist_track_details$deezer_id)),]
  for (i in 1:nrow(Tracks3)){
    variables_musique = Tracks3[i,][,c("acousticness","danceability","energy","liveness","loudness","sp
    ecart = (variables_musique - moyenne_variables)^2/moyenne_variables
    musique_recommandee[i] = sum(abs(ecart)) + 2*(nrow(playlist_track_details) - sum(Tracks3[i,]$name ==
  }
  indice_recommandation = order(musique_recommandee)[1:5]
  data.frame(Titre = Tracks3[indice_recommandation,]$title,Album = Tracks3[indice_recommandation,]$albu
}

playlist_test = playlist[1507,]
playlist_test = numero(playlist_test)
playlist_track_details = Tracks2 %>% filter(deezer_id %in% playlist_test[seq(1,length(playlist_test),2)]


Recommandation_sortie = recommendation(playlist[1507,])
```

Sur un exemple concret on prend une playlist d'entrée et on regarde les recommandations obtenues :

Playlist en entrée :

```
knitr::kable(playlist_track_details[,c("album_title","artist_name","genre","name")])
```

album_title	artist_name	genre	name
Émotions	Jul	trap francais	Jul
Émotions	Jul	trap francais	Jul
Cousine	Jul	trap francais	Jul
Émotions	Jul	trap francais	Jul
My World	Jul	trap francais	Jul
Indeed	Sugar Daddy	NA	Sugar Daddy
Ipséité	Damso	french hip hop	Damso
Ipséité	Damso	french hip hop	Damso

Recommandation en sortie :

```
knitr::kable(Recommandation_sortie)
```

Titre	Album	Artiste	Genre
Je trouve pas le sommeil	Je trouve pas le sommeil	Jul	trap francais
Lady	Je tourne en rond	Jul	trap francais
Dans le futur	My World	Jul	trap francais
Où je vais	Émotions	Jul	trap francais
Nique-le	Je trouve pas le sommeil	Jul	trap francais

Conclusion

Dans ce rapport, nous avons dû traiter les données puis fusionner le tout ensemble pour avoir le plus d'informations possibles pour chaque musique. Cela nous a permis de faire un grand nombre d'analyse comme des graphiques, des cartes ou des classifications. Nous avons également pu créer un algorithme de recommandation qui s'apparenterait à une minimisation de la différence des variables par rapport à la playlist. On peut cependant souligner que les données qu'on a ne sont pas complètes et qu'elles pourraient être enrichies avec toute la base de donnée de Deezer.