Projet R avancé

Yann KIBAMBA Nabil LOUDAOUI Fatima Ezzahrae GOUARAB

2023-04-24

```
#install.packages("webshot")
webshot::install_phantomjs()
```

library

```
# install.packages("mongolite")
# install.packages("rvest")
# install.packages("DBI")
# install.packages("RSQLite")
library(rvest)
library(httr)
library(dplyr)
library(tidyverse)
library(jsonlite)
library(mongolite)
library(DBI)
library(FactoMineR)
library(RSQLite)
library(plotly)
library(leaflet)
library(sf)
```

Introduction

New York est une ville emblématique située sur la côte Est des États-Unis. Elle est connue pour ses gratte-ciels, ses quartiers animés, ses attractions touristiques, ses musées renommés, ses restaurants de renommée mondiale. C'est également un centre financier et commercial majeur, ainsi qu'un carrefour de la culture, de l'art et du divertissement. La ville est souvent considérée comme l'une des destinations les plus passionnantes et les plus dynamiques au monde. C'est chaque année l'Etat de New York accueil près de 60 millions de touristes. Cependant, derrière cette prospérité se cache un niveau de criminalité relativement élevé. Les inégalités socio-économiques et raciales ont un impact important sur la criminalité dans cet État.

Dans ce projet, nous allons utiliser différentes sources de données portant sur cet Etat afin de pouvoir se mettre dans la peau d'un touriste.

Nous recueillerons d'une part des informations sur le niveaux de criminalité dans les différents quartiers, mais aussi des informations sur la démographie et aussi des données relatives à la restauration et l'hébergement sur place.

Tout d'abord, nous utiliserons l'API **NYPD** qui recense toutes les plaintes de l'état de New York. Cela nous permettra d'obtenir une vue d'ensemble des différents types de crimes commis dans les différents quartiers de l'État. L'API **YELP FUSION** sera utile pour obtenir les informations quant aux structures sur place. Nous utiliserons également le web scrapping pour collecter des données sur la démographie des différents quartiers de l'état de New York. Cela nous permettra d'examiner les corrélations potentielles entre la démographie d'un quartier et son niveau de criminalité.

Dans un dernier temps nous explorerons une base issu du serveur MongoAtlas.

A noter que dans ce projet, seul les quartiers de Manhattan, Brooklyn, du Bronx et du Queens seront étudiés.

Partie 1: Récupération des données & Présentation des données

API jeu de données NYPD Complaint Data Historic

La ressource contient un peu plus de 7 millions de plaintes recensées dans l'Etat de New York. Nous en avons conserver 10 000 pour le projet afin d'avoir une assez bonne représentativité. Le dataset est assez précis sur les plainte, de ce fait il contient 41 colonnes. En autre, l'identifiant de la plainte, l'horaire et coordonnées de cette dernière en plus de celle de l'évènement, les différentes caractéristiques du suspect et celle de la victime et bien d'autres encore type de données. L'extraction de données pour cette API ne nécessite pas de clé.

```
url <- "https://data.cityofnewyork.us/resource/qgea-i56i.json?$limit=10000"

response <- GET(url)

data_NYPD <- content(response, "text") %>%
  fromJSON(flatten = TRUE)

# data_NYPD
```

Présentation des variables qui seront utilisé par la suite:

```
table(data_NYPD$law_cat_cd)
table(data_NYPD$boro_nm)
table(data_NYPD$crm_atpt_cptd_cd)
table(data_NYPD$prem_typ_desc)
```

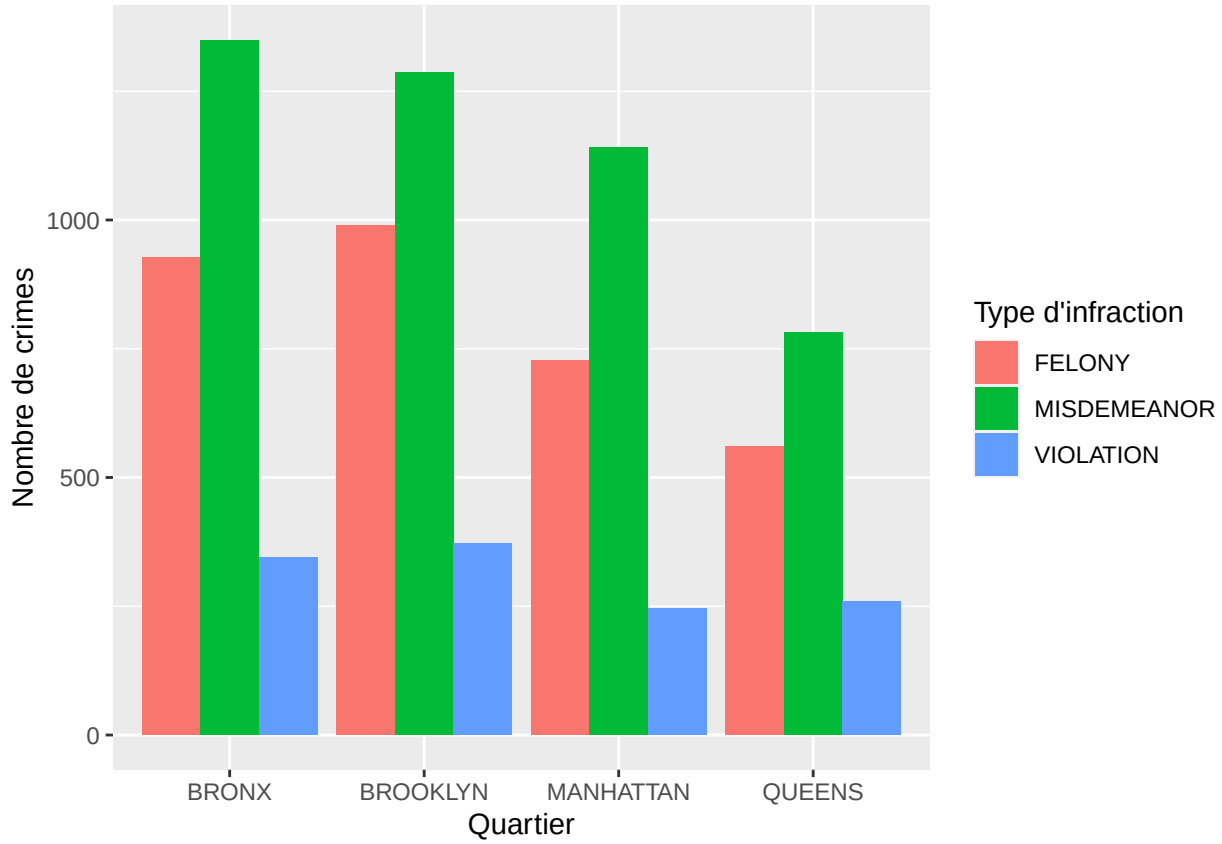
law_cat_cd –> Type de d'infraction (Meutre, délit mineur, Viol) boro_nm –> Nom du quartier de l'Etat de New York (BRONX, BROOKLYN,MANHATTAN,QUEENS,STATEN ISLAND) crm_atpt_cptd_cd –> Indicateur indiquant si le crime a été commis ou tenté avec succès, mais a échoué ou a été interrompu prématurément. (ATTEMPTED COMPLETED) prem_typ_desc –> Description précise des locaux; épicerie, résidence, rue, etc. longitude –> Longitude du lieu de l'évènement latitude –> Latiude du lieu de l'évènement

```
# Création de la table pour l'exploitation des crimes
NY_crime <- data_NYPD %>%
  select("law_cat_cd", "boro_nm","crm_atpt_cptd_cd", "prem_typ_desc","longitude", "latitude")
```

Quelques visualisations sur la base de données:

```
crime_borough <- NY_crime %>%
  group_by(boro_nm, law_cat_cd) %>%
  summarise(count= n())
```

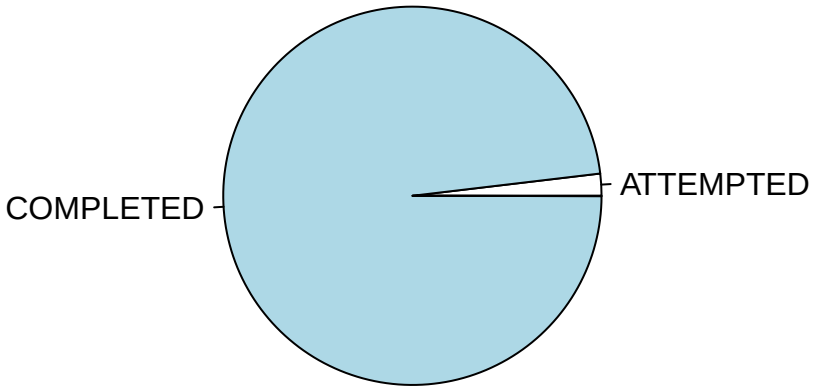
```r
crime_borugh <- na.omit(crime_borough)

crime_borough <- crime_borough %>%
  filter(boro_nm != "STATEN ISLAND")

ggplot(crime_borough, aes(x = boro_nm, y = count, fill = law_cat_cd)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Quartier", y = "Nombre de crimes", fill = "Type d'infraction")
```



Pour notre échantillon de 10000 plaintes, nous avons globalement un nombre plus élevé provenant de Brooklyn, puis de Manhattan, puis du Bronx et enfin du Queens. Les délits mineurs sont en majorités, c'est ensuite les meutres et enfin les viols.

Pourcentage de tentatives d'effraction et effraction réussis

```r
# Pourcentage
tentative_borough <- NY_crime %>%
  group_by(crm_atpt_cptd_cd) %>%
  summarise(count= n())

tentative_borough <- tentative_borough %>%
  mutate(fraction = count / sum(count))


# Visualisation
pie(tentative_borough$fraction, labels = c("ATTEMPTED","COMPLETED"))
```



Dans l'ensemble, la plupart des effractions s'avère au final réussis pour le suspect.

API YELP FUSION

Yelp Fusion permet d'extraire des données sur les entreprises enregistrées sur la plateforme Yelp pour la ville de New York, telles que les informations sur l'emplacement, les avis et les évaluations, les heures d'ouverture, les photos et plus encore. Ces données peuvent être utilisées pour analyser et comprendre les tendances et les préférences des consommateurs, pour évaluer la concurrence dans un secteur donné ou pour aider à prendre des décisions commerciales éclairées. Dans notre cas nous avons extrait des données relatives aux restaurants et hôtels pour les 4 quartiers de New York représentés dans l'étude.

Pourl'accès aux données une clé d'API est nécessaire.

```
# On insére votre clé d'API Yelp Fusion ici
api_key <- "aREyxLNOuhv-uFXyotVdWNS5woLJh5ABMMgx9a7CYJS_4QNppLFzxBAjQ3idnH1vmQcMX9qC6JZZB_Y7-f-X5bvK9ra
```

Ci-dessous l'extraction des restaurants pour Manhattan, Brooklyn, le Queens et le Bronx. ### Restaurant New York

```
# On veut  récupérer une liste de restaurants dans un rayon de 20000
# Définir les paramètres de la requête
params_restaurant_brooklyn <- list(
  term = "restaurant",
  location = "Brooklyn",
  radius = 20000,
  limit =35
```

```
)

# Envoyer la requête et récupérer les données
response_restaurant_brooklyn <- GET("https://api.yelp.com/v3/businesses/search",
                add_headers(Authorization = paste("Bearer", api_key)),
                query = params_restaurant_brooklyn)
data_restaurant_brooklyn <- fromJSON(content(response_restaurant_brooklyn, "text"), flatten = TRUE)$busi
data_restaurant_brooklyn <-as.data.frame(data_restaurant_brooklyn)
data_restaurant_brooklyn <- data_restaurant_brooklyn%>%
  mutate("Quartier" = "Brooklyn")

#################################################################

params_restaurant_queens <- list(
  term = "restaurant",
  location = "Queens",
  radius = 20000,
  limit =35
)

# Envoyer la requête et récupérer les données
response_restaurant_queens <- GET("https://api.yelp.com/v3/businesses/search",
                add_headers(Authorization = paste("Bearer", api_key)),
                query = params_restaurant_queens)
data_restaurant_queens <- fromJSON(content(response_restaurant_queens, "text"), flatten = TRUE)$business
data_restaurant_queens <-as.data.frame(data_restaurant_queens)
data_restaurant_queens <- data_restaurant_queens%>%
  mutate("Quartier" = "Queens")

#########################################################

params_restaurant_Manhattan <- list(
  term = "restaurant",
  location = "Manhattan",
  radius = 20000,
  limit =35
)

# Envoyer la requête et récupérer les données
response_restaurant_Manhattan <- GET("https://api.yelp.com/v3/businesses/search",
                add_headers(Authorization = paste("Bearer", api_key)),
                query = params_restaurant_Manhattan)
data_restaurant_Manhattan <- fromJSON(content(response_restaurant_Manhattan, "text"), flatten = TRUE)$bu
data_restaurant_Manhattan <-as.data.frame(data_restaurant_Manhattan)
data_restaurant_Manhattan <- data_restaurant_Manhattan %>%
  mutate("Quartier" = "Manhattan")


################################################

params_restaurant_Bronx <- list(
  term = "restaurant",
  location = "Bronx",
```

```
  radius = 20000,
  limit =35
)

# Envoyer la requête et récupérer les données
response_restaurant_Bronx <- GET("https://api.yelp.com/v3/businesses/search",
                add_headers(Authorization = paste("Bearer", api_key)),
                query = params_restaurant_Bronx)
data_restaurant_Bronx <- fromJSON(content(response_restaurant_Bronx, "text"), flatten = TRUE)$business
data_restaurant_Bronx <-as.data.frame(data_restaurant_Bronx)
data_restaurant_Bronx <- data_restaurant_Bronx %>%
  mutate("Quartier" = "Bronx")
```

A présent fusionnons les bases de restaurant des Quartiers.

```
# Fusion des bases de Quartiers
restaurants <- rbind(data_restaurant_Bronx, data_restaurant_Manhattan,data_restaurant_brooklyn,data_res
restaurants <- restaurants %>%
  select(name, categories, rating, price, review_count, phone,transactions,
         location.address1,location.zip_code,coordinates.longitude,coordinates.latitude, Quartier)
restaurants <- na.omit(restaurants)

# Création du type "restaurant"
restaurants <- restaurants %>%
  mutate("Type" = "restaurant")
```

Hotel New york

```
# On veut  récupérer une liste de Hotels dans un rayon de 20000
# Définir les paramètres de la requête
params_Hotel_brooklyn <- list(
  term = "Hotel",
  location = "Brooklyn",
  categorie = "Hotel",
  radius = 20000,
  limit =35
)

# Envoyer la requête et récupérer les données
response_Hotel_brooklyn <- GET("https://api.yelp.com/v3/businesses/search",
                add_headers(Authorization = paste("Bearer", api_key)),
                query = params_Hotel_brooklyn)
data_Hotel_brooklyn <- fromJSON(content(response_Hotel_brooklyn, "text"), flatten = TRUE)$business
data_Hotel_brooklyn <-as.data.frame(data_Hotel_brooklyn)
data_Hotel_brooklyn <- data_Hotel_brooklyn%>%
  mutate("Quartier" = "Brooklyn")

################################################################

params_Hotel_queens <- list(
  term = "Hotel",
```

```
  location = "Queens",
  radius = 20000,
  categorie = "Hotel",
  limit =35
)

# Envoyer la requête et récupérer les données
response_Hotel_queens <- GET("https://api.yelp.com/v3/businesses/search",
                add_headers(Authorization = paste("Bearer", api_key)),
                query = params_Hotel_queens)
data_Hotel_queens <- fromJSON(content(response_Hotel_queens, "text"), flatten = TRUE)$business
data_Hotel_queens <-as.data.frame(data_Hotel_queens)
data_Hotel_queens <- data_Hotel_queens %>%
  mutate("Quartier" = "Queens")

#########################################################

params_Hotel_Manhattan <- list(
  term = "Hotel",
  location = "Manhattan",
categorie = "Hotel",
  radius = 20000,
  limit =35
)

# Envoyer la requête et récupérer les données
response_Hotel_Manhattan <- GET("https://api.yelp.com/v3/businesses/search",
                add_headers(Authorization = paste("Bearer", api_key)),
                query = params_Hotel_Manhattan)
data_Hotel_Manhattan <- fromJSON(content(response_Hotel_Manhattan, "text"), flatten = TRUE)$business
data_Hotel_Manhattan <-as.data.frame(data_Hotel_Manhattan)
data_Hotel_Manhattan <- data_Hotel_Manhattan%>%
  mutate("Quartier" = "Manhattan")


#################################################

params_Hotel_Bronx <- list(
  term = "Hotel",
  location = "Bronx",
  radius = 20000,
categorie = "Hotel",
  limit =35
)

# Envoyer la requête et récupérer les données
response_Hotel_Bronx <- GET("https://api.yelp.com/v3/businesses/search",
                add_headers(Authorization = paste("Bearer", api_key)),
                query = params_Hotel_Bronx)
data_Hotel_Bronx <- fromJSON(content(response_Hotel_Bronx, "text"), flatten = TRUE)$business
data_Hotel_Bronx <-as.data.frame(data_Hotel_Bronx)
data_Hotel_Bronx <- data_Hotel_Bronx%>%
  mutate("Quartier" = "Bronx")
```

A présent fusionnons les bases d'hôtel des Quartiers.

```
# Fusion des bases d'hôtels des différents quatiers
Hotels <- rbind(data_Hotel_Bronx, data_Hotel_Manhattan,data_Hotel_brooklyn,data_Hotel_queens)
Hotels <- Hotels %>%
  select(name, categories, rating, price, review_count, phone,transactions,
         location.address1,location.zip_code,coordinates.longitude,coordinates.latitude, Quartier)
Hotels <- na.omit(Hotels)

# Création du type Hotel
Hotels <- Hotels %>%
  mutate("Type" = "hotel")
```

Ainsi, toutes les informations seront contenus dans une base nommée tourism afin de mieux aiguiller ces derniers.

```
tourism <- rbind(restaurants,Hotels)
# tourism
```

Croisement de données de données

Ajoutons le taux de crime d'un quartier à la base de données tourism. Le taux de crime sera représenté par la proportion de FELONY et VIOLATION sur l'ensemble des infractions dans le quartier.

```
taux_crimi <- NY_crime%>%
  group_by(boro_nm)%>%
  filter(((boro_nm == "BRONX" | boro_nm =="BROOKLYN" | boro_nm =="MANHATTAN" | boro_nm =="QUEENS") & lau
  summarise(count= n())%>%
  mutate(taux = count/sum(count))%>%
  select(boro_nm, taux)

taux_crimi <- na.omit(taux_crimi)
```

Ajout à la base tourism des différents taux de criminalité trouvés.

```
tourism <- tourism %>%
  mutate(crime = case_when(
    Quartier == "Bronx" ~ taux_crimi$taux[1],
    Quartier == "Brooklyn" ~ taux_crimi$taux[2],
    Quartier == "Manhattan" ~ taux_crimi$taux[3],
    Quartier == "Queens" ~ taux_crimi$taux[4],
    TRUE ~ 1
  ))

# Renommage de variables dans la base torusim
colnames(tourism)[colnames(tourism) == "coordinates.longitude"] <- "longitude"
colnames(tourism)[colnames(tourism) == "coordinates.latitude"] <- "latitude"
colnames(tourism)[colnames(tourism) == "coordinates.latitude"] <- "latitude"
colnames(tourism)[colnames(tourism) == "location.address1"] <- "addresse"
colnames(tourism)[colnames(tourism) == "location.zip_code"] <- "zip_code"
#names(tourism)
```

Maintenant que la base de données tourism est prête à l'exploitation, nous la placerons dans un serveur SQL. Les variables categories et transactions disposent de valeurs de type "list", pour cette raison il ne sera pas possible de les employés dans cette partie SQL. Ainsi ci-dessous nous répondrons à différentes questions qu'un voyageur à New York pourrait se demander. Connexion:

```
tourism2 <- subset(tourism, select = -c(categories, transactions))


con <- dbConnect(SQLite(), dbname="mydatabase.sqlite")

table <- dbWriteTable(con, "mytable", tourism2, overwrite = TRUE)


db <- tbl(con,"mytable")
```

Requête: Sélection des restaurants ayant plus de 2000 commentaires.

```
query <- "SELECT name FROM mytable WHERE review_count>2000 "
df <- dbGetQuery(con, query)
df
```

```
##                          name
## 1         La Grande Boucherie
## 2                   The Smith
## 3            Anytime Kitchen
## 4           Jack's Wife Freda
## 5             Jacob's Pickles
## 6                      Amélie
## 7              Joe's Shanghai
## 8               Tanner Smiths
## 9                   Scarpetta
## 10                   L'Artusi
## 11                   Cookshop
## 12                        Bea
## 13                  Juliana's
## 14                Sugar Freak
## 15 SriPraPhai Thai Restaurant
```

Selection des quartiers qui ont un taux de criminalités faible

```
query2 <- "SELECT Quartier, AVG(crime) as taux_criminalite
                            FROM mytable
                            GROUP BY Quartier
                            HAVING taux_criminalite < 0.2
                            ORDER BY taux_criminalite ASC"
df2 <- dbGetQuery(con, query2)
df2
```

```
##   Quartier taux_criminalite
## 1    Bronx         0.166623
```

Touriste cherchant un endroit securisé , agréable pour manger avec un budget modéré en se basant sur le résultat de la requête 2.

```
# NB: Dans les données que nous  récupérées, la colonne "price" contient des informations sur la fourch

query3 <-"SELECT name, rating, price, review_count
FROM mytable
WHERE Type = 'restaurant'
AND rating >= 4.5
AND price = '$$'
AND Quartier = 'Bronx'
ORDER BY review_count DESC
LIMIT 10"

df3 <- dbGetQuery(con, query3)
df3
```

```
##                         name rating price review_count
## 1        Antonio’s Trattoria    4.5    $$         1027
## 2         Taqueria Tlaxcalli    4.5    $$          743
## 3            Chocobar Cortes    4.5    $$          385
## 4             The Honey Well    4.5    $$          378
## 5       The Rail House 10803    4.5    $$          135
## 6               ÇKA KA QËLLU    4.5    $$          126
## 7       Archie’s Tap & Table    4.5    $$          108
## 8 Zhongzhong Noodles - Bronx    4.5    $$           59
```

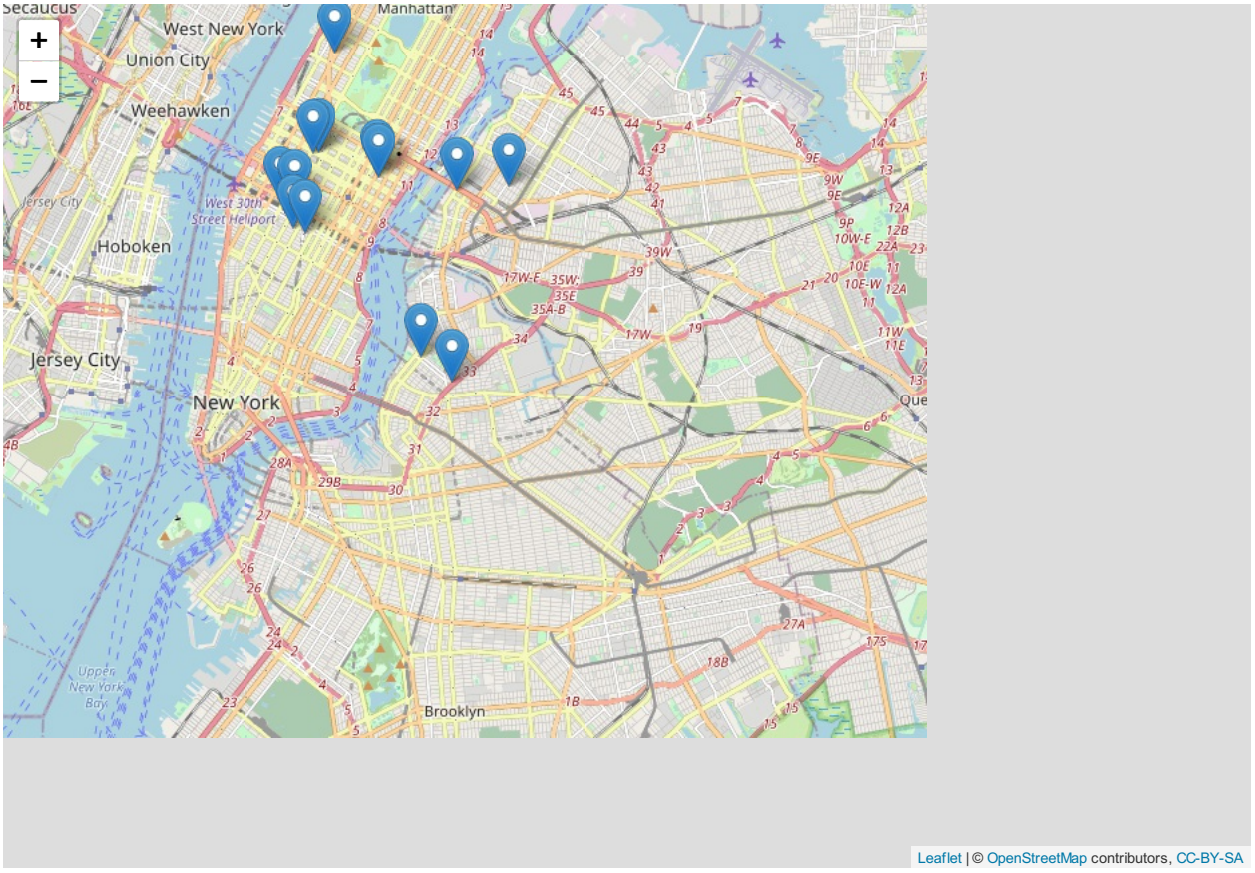
Toujours dans le cas d'un touriste qui cherche un hebergement dans un hotel , qui sera à la fois proche du Bronx et populaire dans le sens où (review_count est elevé)

```
query4<- "SELECT name, rating, price, review_count, addresse || ', ' || zip_code AS adresse_complete , l
FROM mytable
WHERE Type = 'hotel'
AND price = '$$'
AND SQRT(POWER(69.1 * (latitude - 40.84480), 2) + POWER(53 * (longitude + 73.86480), 2)) <= 10
ORDER BY review_count DESC, rating DESC
LIMIT 15
"

df4 <- dbGetQuery(con, query4)
head(df4)
```

```
##                                 name rating price review_count
## 1                  Ace Hotel New York    4.0    $$          762
## 2              The Williamsburg Hotel    3.0    $$          401
## 3                         Ravel Hotel    2.5    $$          383
## 4       Hilton Garden Inn Times Square    3.5    $$          379
## 5 citizenM New York Times Square Hotel    4.0    $$          249
## 6   Renaissance New York Midtown Hotel    4.0    $$          193
##           adresse_complete latitude longitude
## 1      20 W 29th St, 10001 40.74578 -73.98829
## 2      96 Wythe Ave, 11249 40.72151 -73.95870
## 3 8-08 Queens Plz S, 11101 40.75391 -73.94941
## 4       790 8th Ave, 10019 40.76121 -73.98651
## 5     218 W 50th St, 10019 40.76158 -73.98495
## 6     218 W 35th St, 10001 40.75176 -73.99109
```

Ainsi via les requêtes nous avons reccueilli un liste d'hôtels populaire à proximité du Bronx et des restaurant à prix modéré non loin du Bronx. Ci-dessous une cartographie de la requête 4.

```
# création de la carte centrée sur les coordonnées de New York
map <- leaflet() %>%
  addTiles() %>%
  setView(lng = -73.9857, lat = 40.7484, zoom = 12)

# ajout des marqueurs pour chaque hôtel dans les résultats de la requête
for (i in seq_len(nrow(df4))) {
  name <- df4[i, "name"]
  rating <- df4[i, "rating"]
  address <- df4[i, "adresse_complete"]
  map <- map %>%
    addMarkers(lng = df4[i, "longitude"], lat = df4[i, "latitude"],
               popup = paste("<strong>", name, "</strong><br>",
                             "Rating:", rating, "/ 5<br>",
                             "Address:", address))
}

# affichage
map
```



Une fois nos tâches terminées, nous procédons à la déconnexion au serveur.

```
dbDisconnect(con)
```

Web scrapping

Le web scraping est une technique qui consiste à extraire automatiquement des données structurées ou non-structurées d'un site web en utilisant des outils et des scripts. Dans cette partie nous nous concentrerons à extraire des données démographiques du Bronx, de Manhattan et de Brooklyn via leurs pages web wikipedia.

Bronx

```
url_wiki_bronx <- "https://fr.wikipedia.org/wiki/Bronx"
bronx_html <-  read_html( url_wiki_bronx) # lecture de l'url
tables_bronx <- html_table(bronx_html, fill = TRUE) # toutes les tables de la page

demographie_bronx <- tables_bronx[[5]]
demographie_bronx
```

Données démographique Bronx

```
## # A tibble: 13 x 3
##    Indicateur                              Bronx    ‘États-Unis‘
##    <chr>                                   <chr>    <chr>
##  1 Personnes de moins de 5 ans             7,2 %    6,1 %
##  2 Personnes de moins de 18 ans            24,9 %   22,6 %
##  3 Personnes de plus de 65 ans             12,3 %   15,6 %
##  4 Femmes                                  52,8 %   50,8 %
##  5 Personnes par foyer                     2,84     2,64
##  6 Vétérans                                1,9 %    8,0 %
##  7 Personnes nées étrangères à l’étranger 34,9 %   13,2 %
##  8 Personnes sans assurance maladie        9,4 %    10,2 %
##  9 Personnes avec un handicap              10,3 %   8,6 %
## 10 Revenus annuels                         18 896 $ 29 829 $
## 11 Personnes sous le seuil de pauvreté     28,6 %   12,3 %
## 12 Diplômés du lycée                       71,2 %   87,0 %
## 13 Diplômés de l’université                19,2 %   30,3 %
```

```
langues_bronx <- tables_bronx[[4]]
langues_bronx
```

Langues Bronx

```
## # A tibble: 10 x 4
##    Langues            Bronx   ‘New York‘ ‘États-Unis‘
##    <chr>              <chr>   <chr>      <chr>
##  1 Espagnol           47,16 % 24,56 %    13,05 %
##  2 Anglais            41,74 % 50,96 %    78,95 %
```

```
##  3 Langues africaines 3,49 %  1,07 %     0,33 %
##  4 Français           1,09 %  1,11 %     0,43 %
##  5 Italien            0,72 %  1,00 %     0,22 %
##  6 Chinois            0,49 %  5,92 %     1,05 %
##  7 Arabe              0,43 %  0,80 %     0,35 %
##  8 Créole français    0,36 %  1,41 %     0,27 %
##  9 Tagalog            0,35 %  0,64 %     0,56 %
## 10 Autres             4,17 %  12,53 %    4,79 %
```

Brooklyn

```
url_wiki_brooklyn <- "https://fr.wikipedia.org/wiki/Brooklyn"
brooklyn_html <-  read_html( url_wiki_brooklyn) # lecture de l'url
tables_brooklyn <- html_table(brooklyn_html, fill = TRUE) # toutes les tables de la page

demographie_brooklyn <- tables_brooklyn[[3]]
demographie_brooklyn
```

Données démographique Brooklyn

```
## # A tibble: 13 x 3
##    Indicateur                          Brooklyn ‘États-Unis‘
##    <chr>                               <chr>    <chr>
##  1 Personnes de moins de 5 ans         7,3 %    6,1 %
##  2 Personnes de moins de 18 ans        22,9 %   22,6 %
##  3 Personnes de plus de 65 ans         13,5 %   15,6 %
##  4 Femmes                              52,6 %   50,8 %
##  5 Personnes par foyer                 2,73     2,64
##  6 Vétérans                            1,6 %    8,0 %
##  7 Personnes nées à l’étranger         37,2 %   13,2 %
##  8 Personnes sans assurance maladie    8,8 %    10,2 %
##  9 Personnes avec un handicap          6,0 %    8,6 %
## 10 Revenus annuels                     28 134 $ 29 829 $
## 11 Personnes sous le seuil de pauvreté 20,6 %   12,3 %
## 12 Diplômés du lycée                   80,0 %   87,0 %
## 13 Diplômés de l’université            34,1 %   30,3 %
```

```
langues_brooklyn <- tables_brooklyn[[4]]
langues_brooklyn
```

Langues Brooklyn

```
## # A tibble: 15 x 4
##    Langues            Brooklyn ‘New York‘ ‘États-Unis‘
##    <chr>              <chr>    <chr>      <chr>
##  1 Anglais            53,45 %  50,96 %    78,95 %
##  2 Espagnol           16,62 %  24,56 %    13,05 %
```

```
##  3 Chinois            7,24 %   5,92 %     1,05 %
##  4 Russe              5,33 %   2,49 %     0,30 %
##  5 Créole français    3,07 %   1,41 %     0,27 %
##  6 Yiddish            3,50 %   1,12 %     0,05 %
##  7 Arabe              1,22 %   0,80 %     0,35 %
##  8 Hébreu             1,01 %   0,59 %     0,07 %
##  9 Italien            0,99 %   1,00 %     0,22 %
## 10 Français           0,87 %   1,11 %     0,43 %
## 11 Polonais           0,84 %   0,70 %     0,19 %
## 12 Ourdou             0,84 %   0,60 %     0,14 %
## 13 Langues africaines 0,56 %   1,07 %     0,33 %
## 14 Grec               0,33 %   0,54 %     0,10 %
## 15 Autres             4,13 %   7,13 %     7,50 %
```

Manhattan

```
url_wiki_Manhattan <- "https://fr.wikipedia.org/wiki/Manhattan"
Manhattan_html <-  read_html( url_wiki_Manhattan) # lecture de l'url
tables_Manhattan <- html_table(Manhattan_html, fill = TRUE) # toutes les tables de la page

demographie_Manhattan <- tables_Manhattan[[8]]
demographie_Manhattan
```

Données démographique Manhattan

```
## # A tibble: 8 x 4
##   Indicateur              Manhattan `New York` `États-Unis`
##   <chr>                   <chr>     <chr>      <chr>
## 1 Hommes (%)              47,5      47,4       49,1
## 2 Femmes (%)              52,5      52,6       50,9
## 3 Âge médian              35,7      34,2       35,3
## 4 Moins de 18 ans (%)     16,8      24,2       25,7
## 5 18-64 ans (%)           71        64,1       61,9
## 6 65 ans et + (%)         12,2      11,7       12,4
## 7 Revenu/hab. ($)         42 922    22 402     21 587
## 8 Taux de pauvreté[52] (%) 20       21,2       12,4
```

```
langues_Manhattan <- tables_Manhattan[[10]]
langues_Manhattan
```

Langues Manhattan

```
## # A tibble: 17 x 4
##    Langues            Manhattan `New York` `États-Unis`
##    <chr>              <chr>     <chr>      <chr>
##  1 Anglais            59,73 %   50,96 %    78,95 %
##  2 Espagnol           22,50 %   24,56 %    13,05 %
```

```
##  3 Chinois            5,32 %    5,92 %     1,05 %
##  4 Français           2,36 %    1,11 %     0,43 %
##  5 Coréen             0,89 %    0,97 %     0,38 %
##  6 Russe              0,82 %    2,49 %     0,30 %
##  7 Japonais           0,75 %    0,28 %     0,15 %
##  8 Hébreu             0,69 %    0,59 %     0,07 %
##  9 Allemand           0,67 %    0,27 %     0,34 %
## 10 Langues africaines 0,65 %    1,07 %     0,33 %
## 11 Italien           0,64 %    1,00 %     0,22 %
## 12 Portugais         0,49 %    0,23 %     0,23 %
## 13 Hindi             0,44 %    0,41 %     0,24 %
## 14 Arabe             0,40 %    0,80 %     0,35 %
## 15 Tagalog           0,40 %    0,64 %     0,56 %
## 16 Polonais          0,27 %    0,70 %     0,19 %
## 17 Autres            2,98 %    8,00 %     3,16 %
```

Comparaison de Quartier & Croisement de données

Pour chacun des quartiers sélectionnés de l'Etat de New York nous constitueront une base de données regroupant les informations démographiques, linguistiques et criminelles de chaque quartier. L'idée finale est de pouvoir comparer les quartiers sur des thématiques communes. Ainsi, nous avons pensé qu'une Analyse en Composante Principale serait propice à la réalisation visuelle de la base de données finale. Spécificité de la ville pour une acp

On reprend donc les tables suivantes:

```
# NY_crime
# demographie_brooklyn
# demographie_Manhattan
# demographie_bronx
# langues_brooklyn
# langues_Manhattan
# langues_bronx
```

Dans cette partie nous réutiliserons le taux de criminalité calculé précédemment. Pour rappel dans le projet le taux de criminalité pour un quartier est défini par le nombre de d'infraction grave (VIOLATION & FELONY) sur le nombre total de plainte de ce quartier.

```
taux_crimi <- taux_crimi%>%
rename( "Quartier" = boro_nm)

taux_crimi[1,1] <- "Bronx"
taux_crimi[2,1] <- "Brooklyn"
taux_crimi[3,1] <- "Manhattan"
```

Mise en forme de la table demographie de Brooklyn

```
demographie_brooklyn[2, 1] <- "-18ans"
demographie_brooklyn[3, 1] <- "+65ans"

demographie_brooklyn2 <- demographie_brooklyn %>%
  filter(Indicateur == "-18ans" |Indicateur == "+65ans" | Indicateur == "Femmes" |Indicateur == "Revenus
  select(Indicateur, Brooklyn)
```

```
# Supprimer les $ de la colonne brooklyn
demographie_brooklyn2$Brooklyn <- gsub("\\$", "", demographie_brooklyn2$Brooklyn)

# Supprimer les % de la colonne brooklyn
demographie_brooklyn2$Brooklyn <- gsub("%", "", demographie_brooklyn2$Brooklyn)
demographie_brooklyn2$Brooklyn <- gsub(",", ".", demographie_brooklyn2$Brooklyn)
demographie_brooklyn2$Brooklyn <- gsub(" ", "", demographie_brooklyn2$Brooklyn)

demographie_brooklyn2 <- demographie_brooklyn2 %>%
  mutate(Brooklyn = parse_number(Brooklyn))
```

Mise en forme de la table demographie de Manhattan

```
demographie_Manhattan[2, 1] <- "Femmes"
demographie_Manhattan[4, 1] <- "-18ans"
demographie_Manhattan[6, 1] <- "+65ans"
demographie_Manhattan[7, 1] <- "Revenus annuels"

demographie_Manhattan2 <- demographie_Manhattan %>%
  filter(Indicateur == "-18ans" |Indicateur == "+65ans" | Indicateur == "Femmes" |Indicateur == "Revenus
  select(Indicateur, Manhattan)

# Supprimer les $ de la colonne Manhattan
demographie_Manhattan2$Manhattan <- gsub("\\$", "", demographie_Manhattan2$Manhattan)

# Supprimer les % de la colonne Manhattan
demographie_Manhattan2$Manhattan <- gsub("%", "", demographie_Manhattan2$Manhattan)
demographie_Manhattan2$Manhattan <- gsub(",", ".", demographie_Manhattan2$Manhattan)
demographie_Manhattan2$Manhattan <- gsub(" ", "", demographie_Manhattan2$Manhattan)

demographie_Manhattan2 <- demographie_Manhattan2 %>%
  mutate(Manhattan = parse_number(Manhattan))
```

Mise en forme de la table demographie du Bronx

```
demographie_bronx[2, 1] <- "-18ans"
demographie_bronx[3, 1] <- "+65ans"

demographie_bronx2 <- demographie_bronx %>%
  filter(Indicateur == "-18ans" |Indicateur == "+65ans" | Indicateur == "Femmes" |Indicateur == "Revenus
  select(Indicateur, Bronx)

# Supprimer les $ de la colonne Bronx
demographie_bronx2$Bronx <- gsub("\\$", "", demographie_bronx2$Bronx)
# Supprimer les % de la colonne Bronx
demographie_bronx2$Bronx <- gsub("%", "", demographie_bronx2$Bronx)
demographie_bronx2$Bronx <- gsub(",", ".", demographie_bronx2$Bronx)
demographie_bronx2$Bronx <- gsub(" ", "", demographie_bronx2$Bronx)


demographie_bronx2 <- demographie_bronx2 %>%
  mutate(Bronx = parse_number(Bronx))
```

Jointure des tables des 3 quartiers

```
demographies <- demographie_Manhattan2%>%
  left_join(demographie_bronx2)%>%
  left_join(demographie_brooklyn2)


demographies <- demographies %>%
  pivot_longer(cols = -Indicateur, names_to = "Quartier") %>%
  filter(Quartier %in% c("Bronx", "Brooklyn", "Manhattan")) %>%
  #mutate(value = as.numeric(gsub(",", ".", value))) %>%
  pivot_wider(names_from = Indicateur, values_from = value)
```

Mise en forme des tables linguistiques des 3 quartiers

```
langues_brooklyn2 <- langues_brooklyn%>%
  filter(Langues == "Anglais" | Langues == "Espagnol" | Langues == "Chinois" | Langues == "Français")%>%
  select(Langues,Brooklyn)

langues_Manhattan2 <- langues_Manhattan%>%
  filter(Langues == "Anglais" | Langues == "Espagnol" | Langues == "Chinois" | Langues == "Français")%>%
  select(Langues,Manhattan)

langues_bronx2 <- langues_bronx%>%
  filter(Langues == "Anglais" | Langues == "Espagnol" | Langues == "Chinois" | Langues == "Français")%>%
  select(Langues,Bronx)

language <- langues_Manhattan2%>%
  left_join(langues_bronx2)%>%
  left_join(langues_brooklyn2)

language <- language %>%
  pivot_longer(-Langues, names_to = "Quartier", values_to = "Pourcentage") %>%
  pivot_wider(names_from = "Langues", values_from = "Pourcentage")

language$Anglais <- gsub("%", "", language$Anglais)
language$Espagnol <- gsub("%", "", language$Espagnol)
language$Chinois <- gsub("%", "", language$Chinois)
language$Français <- gsub("%", "", language$Français)

language$Anglais <- gsub(",", ".", language$Anglais)
language$Espagnol <- gsub(",", ".", language$Espagnol)
language$Chinois <- gsub(",", ".", language$Chinois)
language$Français <- gsub(",", ".", language$Français)

language <- language %>%
  mutate(Anglais = parse_number(Anglais))

language <- language %>%
  mutate(Chinois = parse_number(Chinois))

language <- language %>%
  mutate(Espagnol = parse_number(Espagnol))
```

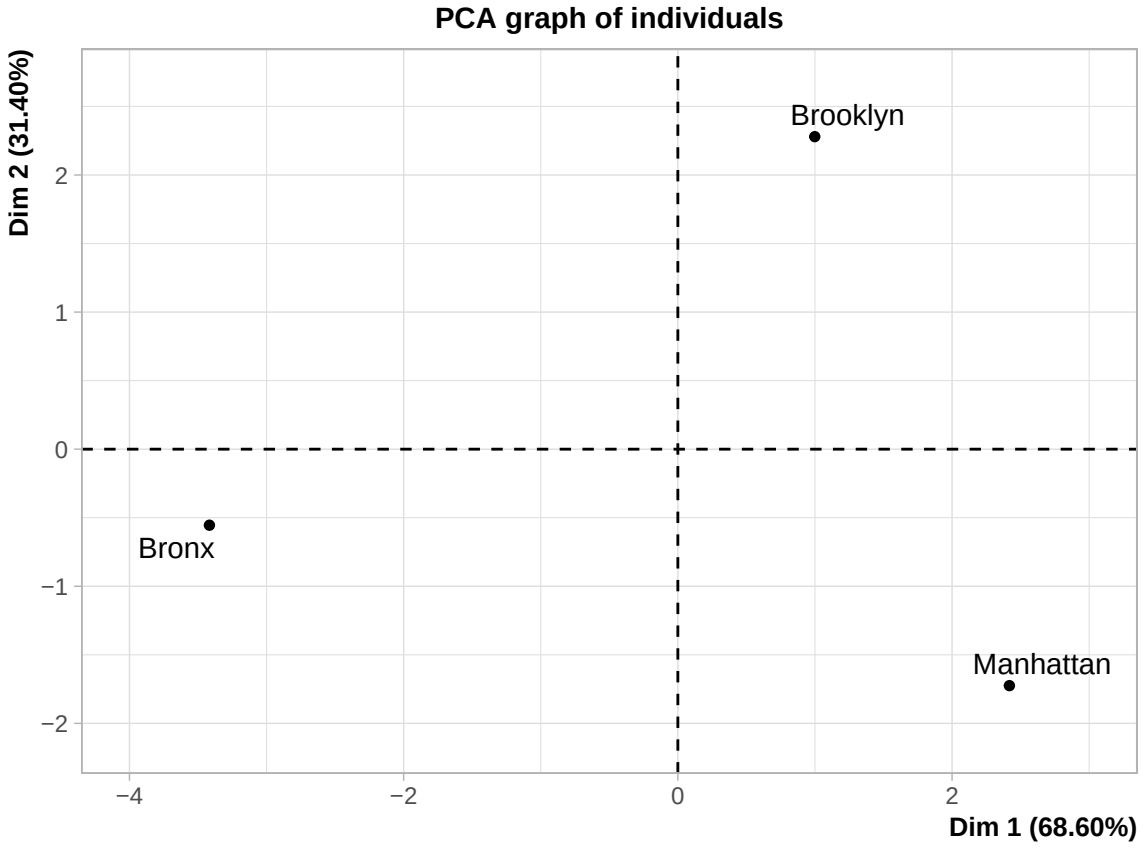
```
language <- language %>%
  mutate(Français = parse_number(Français))
```

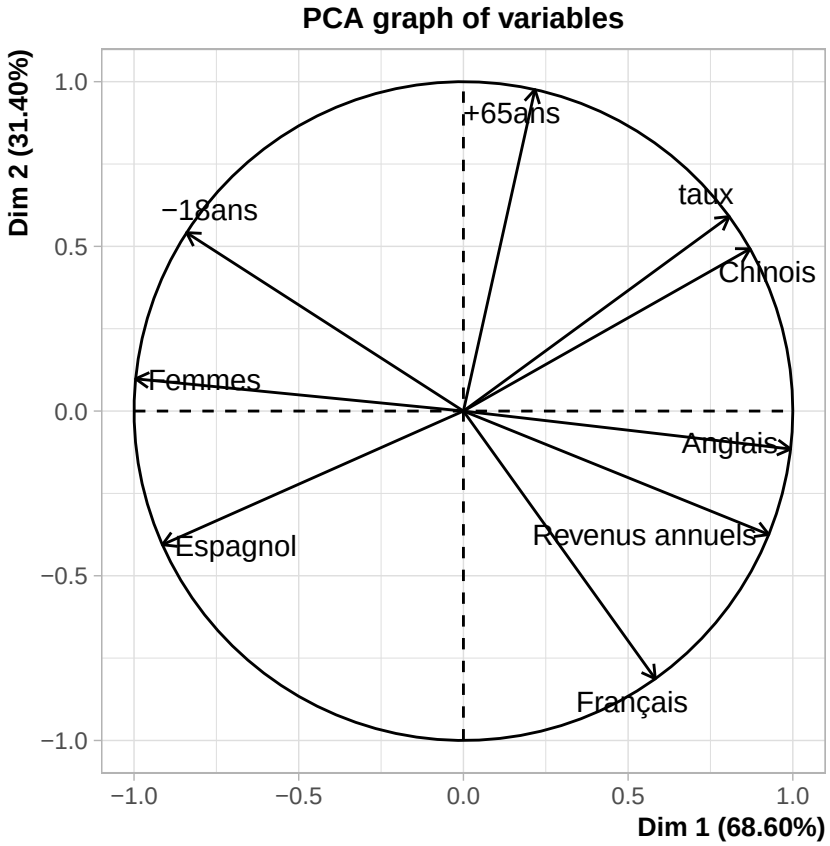
Dans une variable nommée acp, nous joignons toutes nos données manipulées.

```
acp <- merge(language, demographies,by = "Quartier")
acp <- merge(acp, taux_crimi,by="Quartier")
acp
```

```
##    Quartier Anglais Espagnol Chinois Français Femmes -18ans +65ans
## 1     Bronx   41.74    47.16    0.49     1.09   52.8   24.9   12.3
## 2  Brooklyn   53.45    16.62    7.24     0.87   52.6   22.9   13.5
## 3 Manhattan   59.73    22.50    5.32     2.36   52.5   16.8   12.2
##   Revenus annuels      taux
## 1              18 0.1666230
## 2              28 0.3470003
## 3              42 0.2767881
```

```
rownames(acp) <- acp$Quartier
acp <- acp[2:ncol(acp)]
PCA(acp)
```






```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 3 individuals, described by 9 variables
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

Du fait que nous ayons uniquement trois individus, la visualisation de ces derniers dans un plan factoriel est plus sensible aux variables. Ainsi, les 3 quartiers sont assez bien séparés. On a un premier axe expliquant 68% de l'information contenu dans le jeu de données, séparant la proportion de la langue Espagnol contre la langue Anglaise dans le quartier. Quasiment un habitant sur deux dans le Bronx parle espagnol. L'axe 2 met en avant le pourcentage des +65 ans.

Restaurant MONGO DB

Dans cette partie nous procèderons à la connexion à la base de données MongoDB via MongoATLAS. Nous récupérons la base “food” contenant la collection “Nyfood” en utilisant R et la bibliothèque mongo. Cette collection “NYfood” contient des données sur les restaurants de New York. Chaque document de la collection représente un restaurant et inclut des informations telles que l'adresse, le type de cuisine, les notes attribuées à différents moments [A,B,C,D] et un identifiant unique pour le restaurant. Les données sont stockées sous format JSON et peuvent être utilisées pour l'analyse et la visualisation de données.

Nous nous intéresserons principalement aux scores et localisations (gps, adresse, quartiers...) des divers restaurants.

Connexion à la base:

```
url="mongodb://etudiant:ur2@clusterm1-shard-00-00.0rm7t.mongodb.net:27017,clusterm1-shard-00-01.0rm7t.m
mdb = mongo(collection="NYfood", db="food",
            url=url,
            verbose=TRUE)
mdb
```

Nous nous sommes questionné sur les meilleurs lieu de restauration, ainsi nous avons interrogé la bse afin d'obtenir un nombre de 20 restaurants avec les meilleurs notes.

Requete 1 : Quels sont les 20 restaurants (nom, quartier, addresse et score) avec le plus grand score moyen?

Ce code effectue une requête à la base de données MongoDB pour la collection “NYfood” et retourne les 20 restaurants ayant obtenu la meilleure note moyenne, avec leur nom, adresse, score et coordonnées géographiques, en excluant les restaurants dont le quartier est manquant .

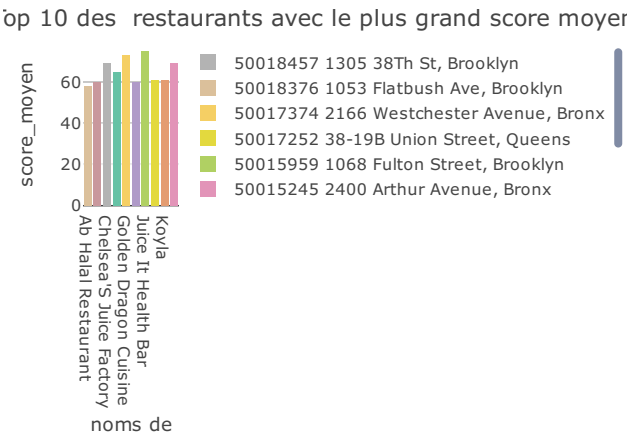
```
qry <- '[  { "$match": { "borough": { "$ne": "Missing" } } },
    { "$unwind": "$grades" },
    { "$group": {
        "_id": {
            "na": "$name",
            "id": "$restaurant_id",
            "bo": "$borough",
            "building": "$address.building",
            "street": "$address.street",
            "loc": "$address.loc.coordinates"
        },
        "sc": { "$avg": "$grades.score" }
    }},
    { "$sort": { "sc": -1 }},
    { "$limit": 20 },
    { "$project": {
        "_id": 0,
        "name": "$_id.na",
        "address": {
            "$concat": [
                "$_id.id", " ", "$_id.building", " ", "$_id.street", ", ", "$_id.bo"
            ]
        },
        "score": "$sc",
        "lon": {"$arrayElemAt": [ "$_id.loc", 0 ]},
        "lat": {"$arrayElemAt": [ "$_id.loc", 1 ]},
        "borough": "$_id.bo"
```

```
    }}
]'

resta <- mdb$aggregate(pipeline = qry)
```

```
##  Found 20 records... Imported 20 records. Simplifying into dataframe...
```

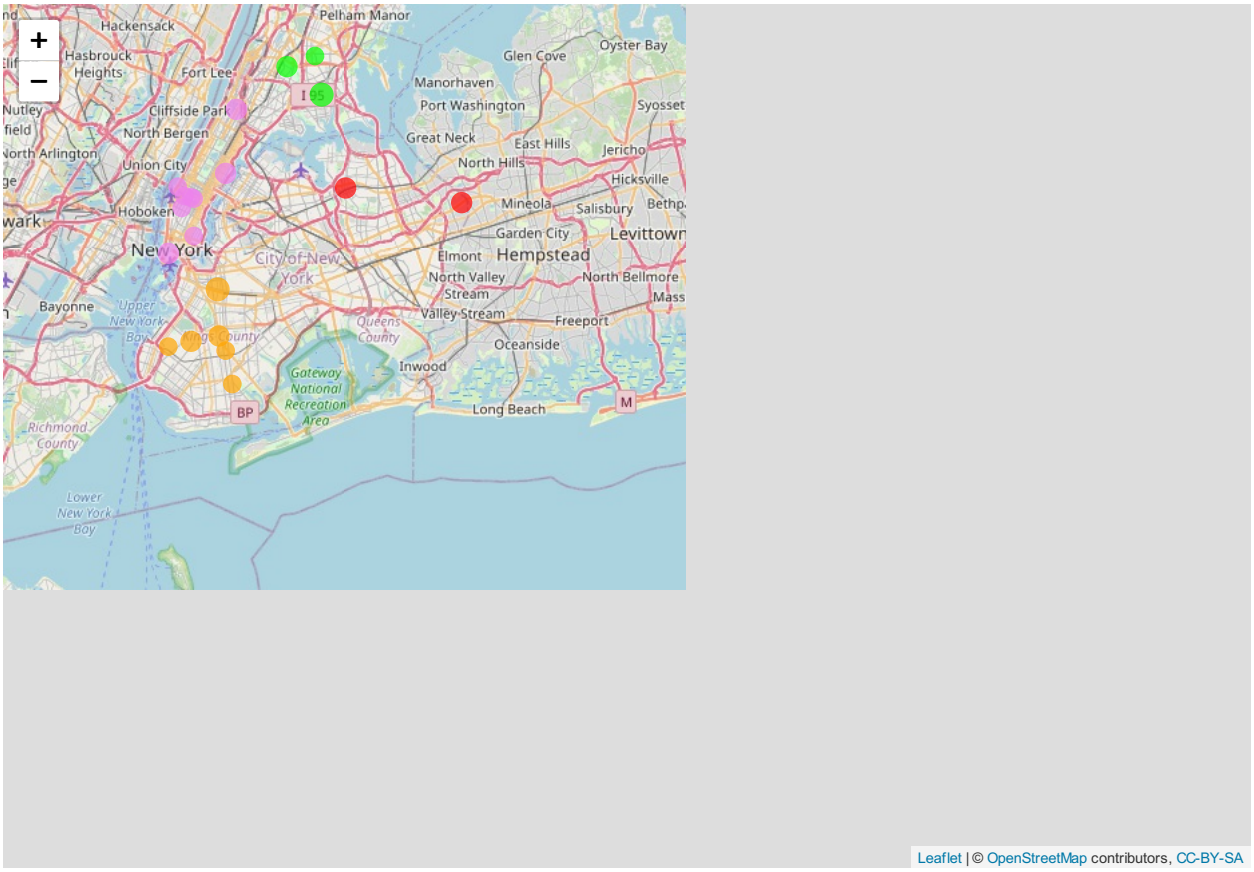
Afin d'avoir un meilleur aperçu des 20 restaurants ayant reçu les meilleurs notes, nous les visualiserons via un barplot et sur une carte interractive. Ci-dessus le barplot avec le score en ordonnée, en abscisse les noms de restaurants.Il est possible de connaitre l'adresse complète du restaurant dans la légende.

```
##Visualisation des 10 restaurants les mieux notés par quartier sur un barplot interactive.

plot <- plot_ly(x = ~resta[1:10,]$name, y = ~resta[1:10,]$score, color = ~resta[1:10,]$address) %>%
  layout(title = "Top 10 des  restaurants avec le plus grand score moyen", xaxis = list(title = "noms d
plot
```




```
#Visualisation des 20 restaurants les mieux notés par quartier sur une carte interactive.

quartiers <- unique(resta$borough)
# Création d'une palette de couleurs pour chaque quartier
pal <- colorFactor(palette = c("green", "orange", "violet", "red"), domain = quartiers)
resta_df <- resta %>% st_as_sf(coords = c("lon", "lat"), crs = 4326)
# Création de la carte avec les marqueurs
```

```
map <- leaflet() %>%
  addTiles() %>%
  addCircleMarkers(data = resta_df,
                   radius = ~sqrt(score),
                   stroke = FALSE,
                   fillOpacity = 0.7,
                   color = ~pal(borough), # Couleur du cercle dépend du quartier
                   popup = ~paste(name, "<br>", address, "<br>", "Score: ", score))

# Affichage de la carte
map
```



On voit que le quartier Manhattan contient le plus de restaurants (8 en total) bien notés, vient ensuite Broklyn qui en contient 6 y compris le restaurant qui a plus haut score. Le Queens en contient 3 sur les 20 avec des scores moyens. Le Bronx en contient également 3 dont le deuxième restaurant le mieux noté. On voit nettement l'avantage de Brooklyn et de Manhattan sur la carte. Cela s'explique sûrement par le fait qu'ils soient les quartiers les plus proche de la ville de **New York**.

Conclusion

Pour ce type d'étude, l'API YELP FUSION est adéquate car riche en information et permet d'obtenir des données en temps réel. (Avantages) Cependant avec cette méthode, on peut être limitée par des quotas et des restrictions d'accès à la base de données en fonction de l'échelle d'extraction (inconvénient). Le raisonnement

est le même concernant l'API **NYPD Complaint Data Historic** où l'utilisateur est dépendant de la mise à jour de la source.

Le serveur MongoDB peut offrir une solution plus évolutive pour stocker et interroger efficacement de grandes quantités de données structurées (avantage), mais peut nécessiter des compétences techniques pour la configuration et la maintenance (inconvénient).

Le web scraping a été efficace pour extraire des données à partir de sources qui ne disposent pas d'une API (directement via la page web wikipedia dans notre cas) ou d'une base de données accessible (avantage), mais peut être plus coûteux et plus complexe en termes de mise en place et de maintenance (inconvénient).

Pour aller plus loin, à l'aide de la richesse de données qu'il est possible d'obtenir, il serait envisageable de concevoir un algorithme dans un but de recommandation pour un touriste avec bien plus de type de lieu.

Sources

Lien API **YELP FUSION**: https://docs.developer.yelp.com/docs Lien API **NYPD Complaint Data Historic** : https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Historic/qgea-i56i/data