Data Science, Séance 4 : Données sur le web + scrapping et API

Etienne Côme

21 novembre 2019

Où trouver des données sur le web

instituts publics : insee, ign, ...

portails open-data : data.iledefrance.fr, data.gouv.fr, ...

sites collaboratifs : wikipedia (dbpedia), openstreetmap, ...

sites specialisés : météo, sports, logement, annonces, ...

réseaux sociaux : twitter, flickR, foursquare, ...

moteur de recherche : google, yahoo, bing, ...

api spécialisées : velib, ...

Où trouver des données sur le web

Jeux de données, déjà mis en forme

instituts publics : insee, ign, . . .

portails open-data : data.iledefrance.fr, data.gouv.fr, . . .

sites collaboratifs : wikipedia (dbpedia), openstreetmap

Où trouver des données sur le web

Jeux de données à mettre en forme

Scrapping

sites specialisés : météo, sports, logement, annonces, ...

API

sites collaboratifs : openstreetmap, ...

réseaux sociaux : twitter, flickR, foursquare, ...

moteur de recherche : google, yahoo, bing, ...

api spécialisées : velib, ...

Scrapping

Extraire des informations spécifiques

d'une ou plusieurs pages web

en vu de constituer un jeu de données

Scrapping, le html

Scrapping, les package RCurl et XML

RCurl (Client URL Request Library)

le web en ligne de commande : get, post, https, ftp, ...

XML

htmlTreeParse, getNodeSet :

Scrapping, les package RCurl et XML

Xpath, extraire des informations d'un arbre DOM

Syntaxe pour se promener dans l'abre dom et en extraire des partie (noeuds, attributs, . . .), plus détails sur w3schools.

Expression	Description
nodename	Selects all nodes with the name "nodename"
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Scrapping, les package RCurl et XML

Xpath, extraire des informations d'un arbre DOM

Syntaxe pour se promener dans l'abre dom et en extraire des partie (noeuds, attributs, . . .), plus détails sur w3schools.

Expression	Description
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element.
//title[@lang]	Selects all the title elements that have an attribute named
//title[@lang='en']	Selects all the title elements that have an attribute named lang with a value of 'en'
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price $>$ 35.00

Library

```
library(XML)
library(RCurl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(rjson)
library(httr)
```

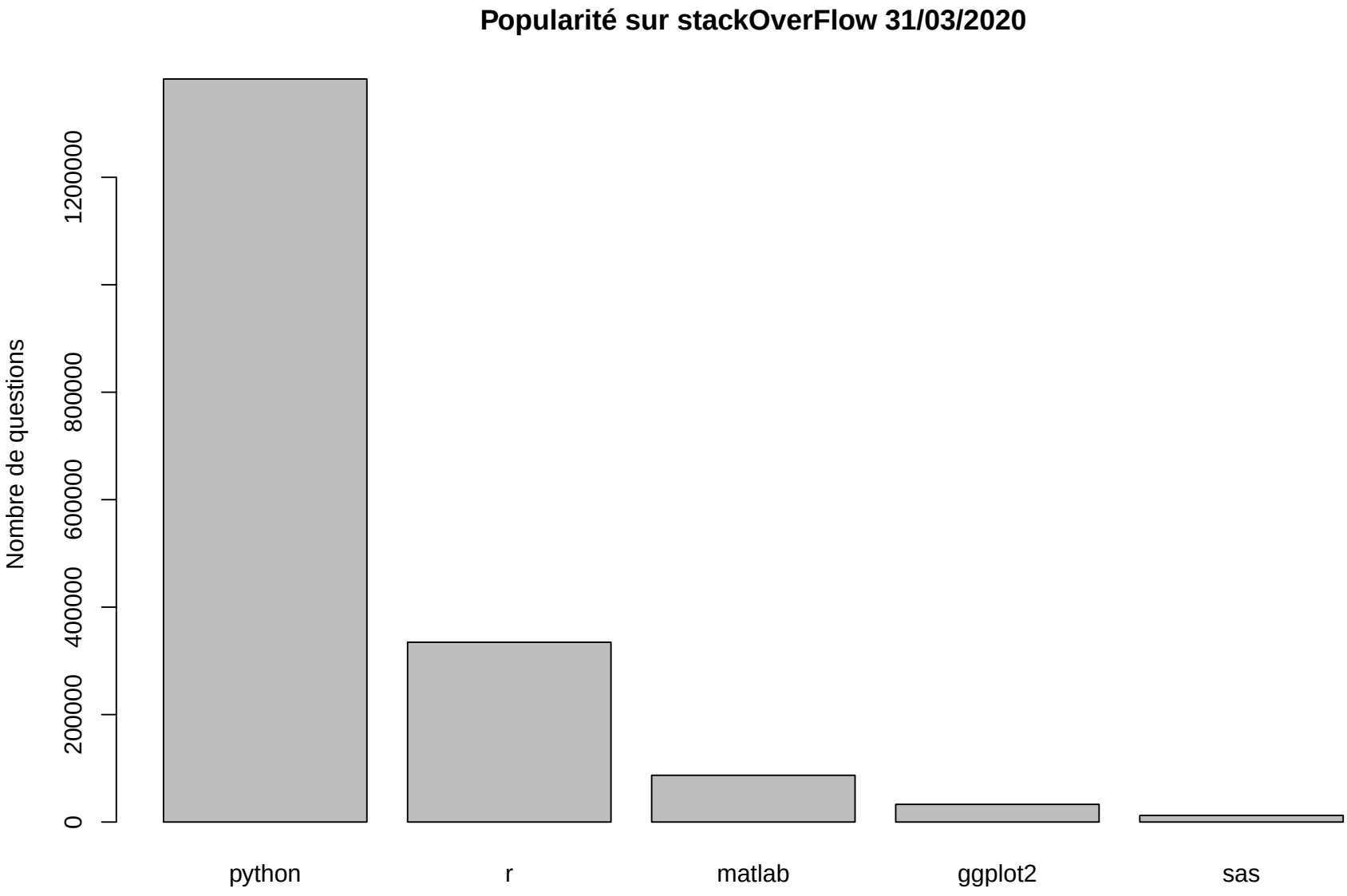
Scrapper stackoverflow.com

Ecrire un script R permettant de scrapper le nombre de question publier sur les sites ayant les tags :
‘python’,‘julia-lang’,‘r’,‘sas’,‘matlab’,‘ggplot2’ et ‘d3.js’. Réaliser un graphique à partir de ces données.

```r
# definition des termes à scrapper
languages=c('python','r','sas','matlab','ggplot2')
# initialisation de la table
stackOF=data.frame(lang=languages,questions=NA)
# boucle sur les termes
for(i in 1:length(languages)){
  # récupérer la page
  base = "https://stackoverflow.com/questions/tagged/"
  res  = getURL(paste(base,stackOF[i,'lang'],sep=''))
  # la parser et récupérer le noeud désiré (xpath)
  resp = htmlTreeParse(res,useInternal=T)
  ns1  = getNodeSet(resp, "//*[@id='mainbar']/div[4]/div/d
  # récupérer la valeure et la nettoyer
  val  = xmlValue(ns1[[1]])
  valclean =  gsub("questions","",val);
  valclean = gsub("[ ,\n,\r]","",valclean)
  stackOF[i,'questions'] = as.numeric(valclean)
}
# faire un graphique
stackOF=stackOF[order(stackOF$questions,decreasing=T),]
```

```r
date = format(Sys.time(), "%d/%m/%Y")
title=paste("Popularité sur stackOverFlow",date)
barplot(stackOF$questions,
        names.arg=stackOF$lang,
        main=title,ylab="Nombre de questions")
```

Scrapper stackOverFlow



Scrapper leboncoin.fr

Ecrire un script R permettant de scrapper le nombre d'annonce de particulier du site dans la catégorie "Jardinage" en région centre.

Scrapper les résultats de ligue 1

Récupérer les dix dernières années de résultats du championnat de france

Scrapper les résultats de ligue 1

```
# récupérer la page et la parser
year = 2018
url_b ="http://www.footballstats.fr/resultat-ligue1-"
res  = getURL(paste(url_b,year,".html",sep=''))
resp = htmlTreeParse(res,useInternal=T)
# récupérer le bon tableau de la page
rest = readHTMLTable(resp)[[2]]
```

Scrapper les résultats de ligue 1

```
# le remettre légèrement en forme
rest = rest[!is.na(rest[,2]),1:4]
names(rest) = c('locaux', 'visiteur','resultat','affluence
rest$locaux=factor(as.character(rest$locaux),
                   levels=unique(rest$locaux))
rest$affluence=as.numeric(as.character(rest$affluence))
rest$visiteurs=factor(as.character(rest$visiteur),
                      levels=unique(rest$locaux))
resm=matrix(unlist(strsplit(as.character(rest$resultat),'-
rest$resultat.locaux=as.numeric(resm[1,])
rest$resultat.visiteurs=as.numeric(resm[2,])
```

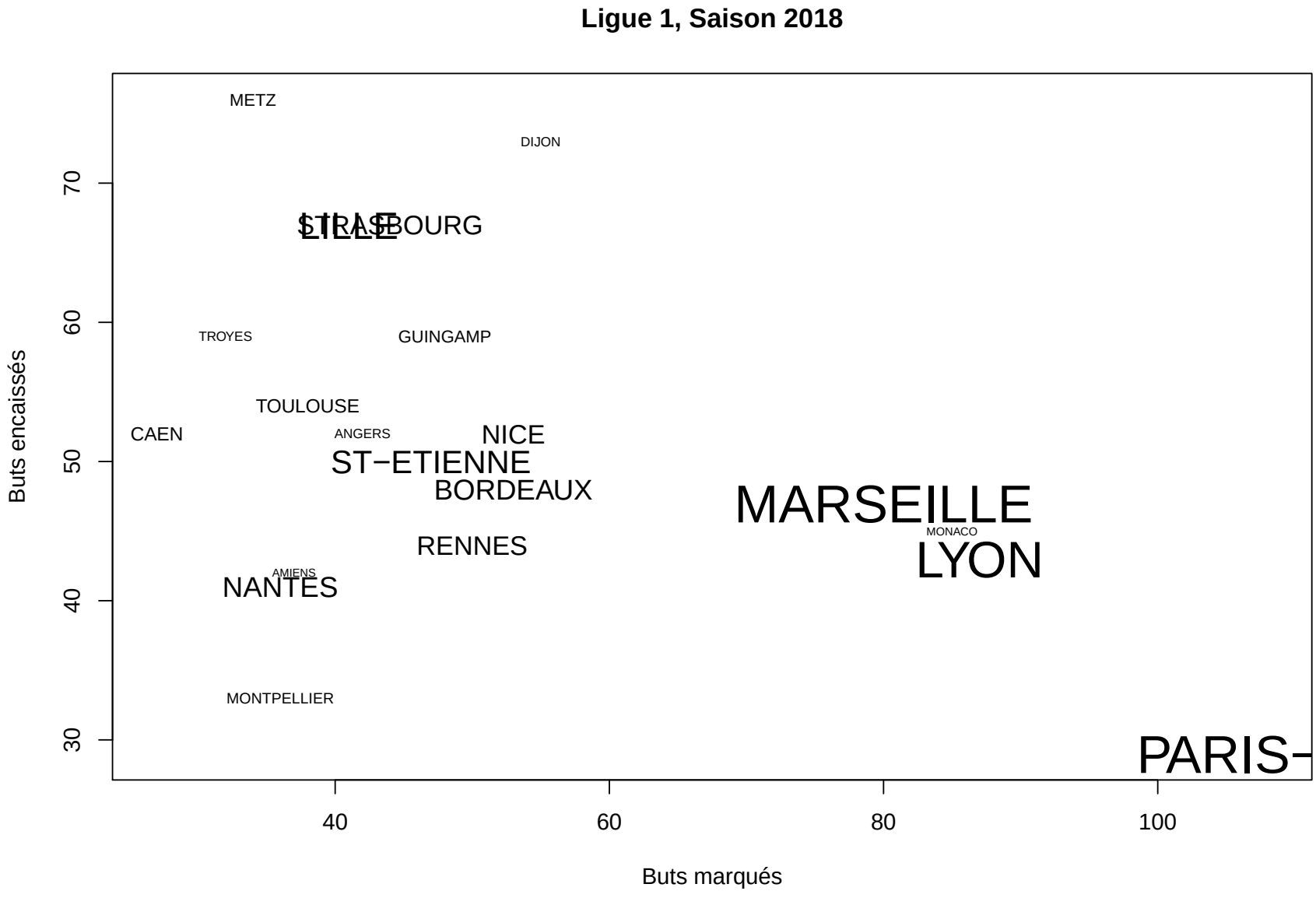
Scrapper les résultats de ligue 1

```
# remise en forme et calcul des totaux de buts marqués / en
resdom = rest  %>% group_by(locaux) %>%
  summarise(Abutdom=sum(resultat.locaux),
            Dbutdom=sum(resultat.visiteurs),
            affdom=mean(affluence,na.rm=TRUE))

resext = rest  %>% group_by(visiteur) %>%
  summarise(Abutext=sum(resultat.visiteurs),
            Dbutext=sum(resultat.locaux),
            affext=mean(affluence,na.rm=TRUE))

res = resdom %>% left_join(resext,by=c("locaux"="visiteur")
  mutate(D=Dbutdom+Dbutext,A=Abutdom+Abutext)
```

Scrapper les résultats de ligue 1

```
# faire un graphique
ti = paste("Ligue 1, Saison",year)
xl = "Buts marqués"
yl = "Buts encaissés"
plot(res$A,res$D,xlab=xl,ylab=yl,col="white",main=ti)
text(res$A,res$D,res$locaux,cex=res$affdom/20000)
```

Scrapper les résultats de ligue 1



API Application Programming Interface

Vélib' et altitude des stations

Utiliser les fichiers http://vlsstats.ifsttar.fr/data/input_Lyon.json et http://vlsstats.ifsttar.fr/data/spatiotemporalstats_Paris.json ainsi que l'api elevation-api.io pour calculer un indicateur de charge moyenne des stations Vélib' et mettre celui-ci en relation avec l'altitude des stations.

Vélib' et altitude des stations

Localisation des stations

```
# récupérer la liste des stations et la mettre en forme
url="http://vlsstats.ifsttar.fr/data/input_Lyon.json"
stationsList=fromJSON(file=url)
data=sapply(stationsList,function(x){
    c(x$number,x$name,x$address,
      x$bike_stands,x$position$lat,x$position$lng)
    })
stations=data.frame(id=as.numeric(data[1,]),name=data[2,],
        adresse=data[3,],   nbdocks=as.numeric(data[4,]),
        lat=as.numeric(data[5,]),long=as.numeric(data[6,]))
```

Vélib' et altitude des stations

API elevation

```
chunk_size = 10
# récupérer les altitudes
base  = "https://elevation-api.io/api/elevation?points="
for (i in 1:ceiling(nrow(stations)/chunk_size)){
system("sleep 0.5")
print(i)
iv    = ((i-1)*chunk_size+1):min((i*chunk_size),dim(stations
latlong = paste0("(",stations[iv,'lat'],",",stations[iv,'lo
query = paste(latlong,collapse=',')
url   = paste(base,query,sep="")
res   = fromJSON(file=url)
stations$alt[iv]=sapply(res$elevations,function(x){x$elevat
}
```

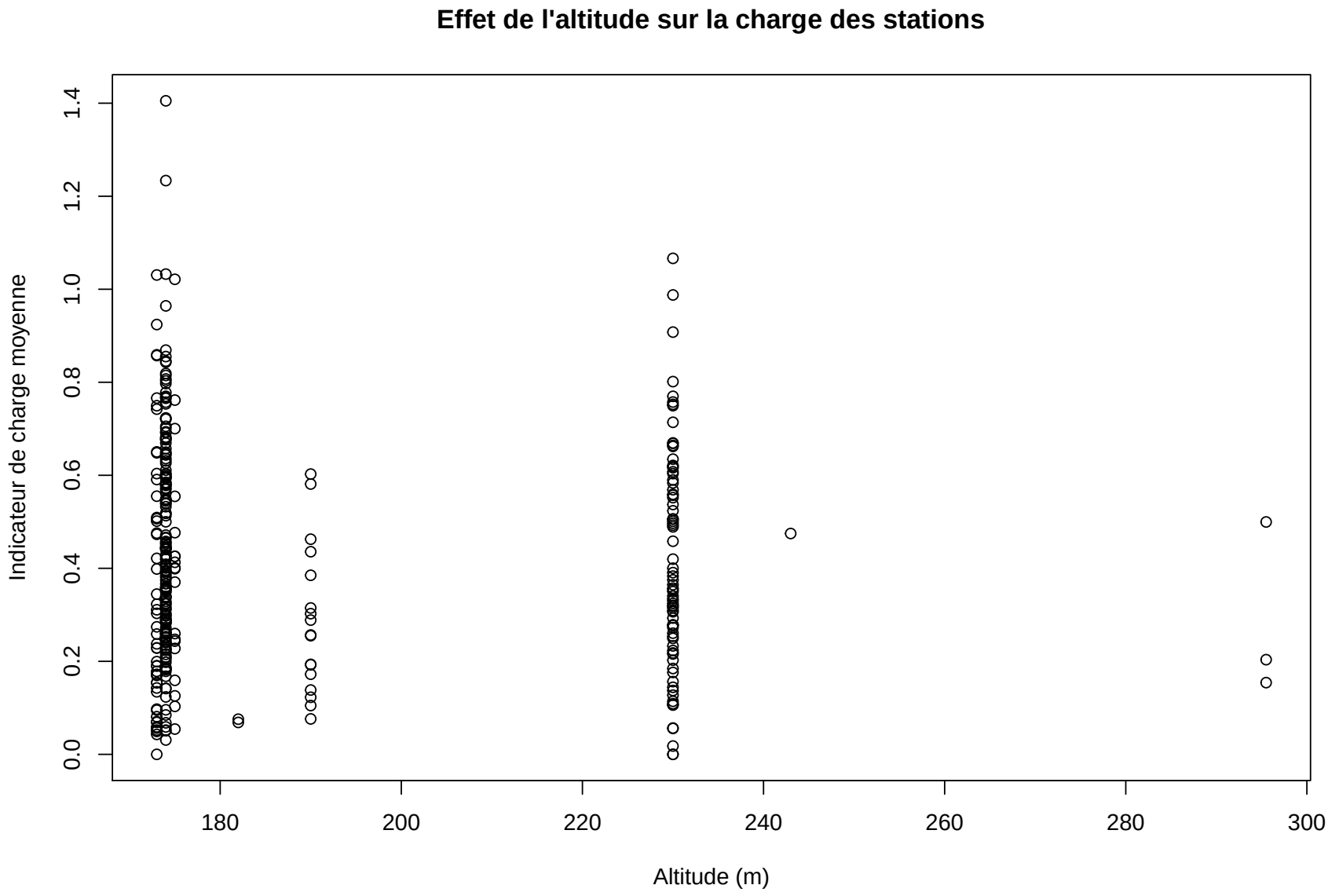
Vélib' et altitude des stations

```
# calculer l'indice de charge moyenne
url = "http://vlsstats.ifsttar.fr/data/spatiotemporalstats_
stationsData = fromJSON(file=url)
res = sapply(stationsData,function(x){
  c(x$'_id', mean(x$available_bikes))})
res = data.frame(t(res),row.names = NULL)
names(res) = c('id','mnbikes')
stations =  stations%>% left_join(res)
```

```
FALSE Joining, by = "id"
```

```
stations$loading = stations$mnbikes/stations$nbdocks
ti = "Effet de l'altitude sur la charge des stations"
yl = "Indicateur de charge moyenne"
xl = "Altitude (m)"
plot(stations$alt,stations$loading,xlab=xl,ylab=yl,main=ti)
```

Vélib' et altitude des stations



API suite

Ecrire une fonction permettant de récupérer le nombre de fan d'un artiste en utilisant l'api deezer.

vous vous servirez de la fonction search et de la fonction artist de cette API.