Analyse de tweets sous R

Tutoriels Tanagra

1. Introduction

Twitter est devenu un instrument incontournable de communication pour tous les acteurs sociaux (Wikipédia). Les hommes politiques, les sportifs, les dirigeants d'entreprises l'utilisent pour donner la primeur de leur actualité, leurs décisions, leurs actions à venir. Il constitue également une plate-forme d'échange qui permet à tout un chacun d'exprimer son opinion en réaction à une annonce ou à un évènement. Des informations, parfois très importantes, transitent ainsi dans tous les sens, tous les jours, sans que nous saisissions toute la portée de ce déluge de textes qui, parfois, semblent peu cohérents.

L'analyse des tweets s'inscrit dans le cadre du text mining à bien des égards. Chaque document est un texte rédigé. Nous pouvons appliquer les techniques de fouille de textes usuelles, notamment en passant à la représentation en sac de mots (bag-of-words). Mais les tweets induisent des particularités. Certaines peuvent enrichir l'analyse. Ainsi, leur longueur est calibrée (du moins en ce qui concerne les messages publics), des caractères spéciaux permettent d'identifier les auteurs (@) et les thématiques (#), les mécanismes de tweet et retweet permettent de suivre la diffusion de l'information. A contrario, d'autres caractéristiques peuvent perturber les analyses. L'espace étant limité, les auteurs utilisent souvent des abréviations, des émoticons pour exprimer des sentiments, et ils ne font pas très attention à l'orthographe. Tout cela engendre du bruit qui peut compliquer notre tâche.

Les tweets véhiculent moult informations potentiellement intéressantes. Leur exploitation est un des axes forts de l'analyse des réseaux sociaux. La société Twitter y contribue en mettant à notre disposition des API (interface de programmation) qui permettent d'accéder par programme aux messages et de réaliser des études statistiques et de data mining. Les perspectives sont nombreuses. Elles vont des statistiques descriptives simples (ex. quels sont les auteurs les plus actifs, quels sont les messages les plus retweetés, etc.) à des investigations plus sophistiquées (ex. quels sont les thèmes émergents, des communautés se sont-elles formées, etc.). Ainsi, les tweets se prêtent à de multiples explorations. Certains outils assez

chatoyants, accessibles en ligne, nous donnent une idée des immenses possibilités qui s'offrent à nous (ex. sentiment viz).

Dans ce tutoriel, nous montrons comment accéder à des messages liés à un thème choisi sur Twitter. Nous initierons une étude relativement basique des propriétés des tweets dans un premier temps. Nous enchaînerons ensuite sur l'exploitation du contenu des messages. Nous travaillerons sous R en nous appuyant sur le package **"twitteR"** de Jeff Gentry qui se révèle particulièrement pratique (Package 'twitteR').

2. Chargement des tweets

Nous devons installer puis charger la librairie "twitteR" pour commencer.

```
#chargement de la librairie

library(twitteR)
```

2.1. Création d'une connexion

L'étape suivante consiste à se connecter sur le serveur de Twitter. Pour cela, nous devons disposer de clés d'authentification que l'on peut obtenir en s'inscrivant sur le site de développement. Le processus n'est pas très complexe, mais il faut être rigoureux. Le tutoriel suivant peut vous guider dans votre démarche d'obtention de ces fameuses clés : https://www.credera.com/blog/business-intelligence/twitter-analytics-using-r-part-1-extract-tweets/

Une fois obtenus, nous spécifions les chaînes de caractères *consumer_key*, *consumer_secret*, *access_token* et *access_secret* dans la commande **setup_twitter_oauth()** :

```
#clés pour la connexion
consumer_key <- "... your consumer key..."
consumer_secret <- "... your consumer secret..."

access_token <- "... your access token ..."
access_secret <- "... your access secret ..."

#Créer une connexion avec  Twitter
setup_twitter_oauth(consumer_key, consumer_secret, access_token, access_secret)
```

Le message "*Using direct authentification*" devrait apparaître dans le console, indiquant le bon déroulement de l'opération.

Ça y est. Nous sommes prêts pour l'extraction des tweets.

2.2. Extraction des tweets

La fonction **searchTwitter()** permet de charger des tweets en ligne. Plusieurs paramètres sont disponibles. Dans notre exemple ci-dessous, nous spécifions le mot clé qui permettra de les sélectionner, nous limitons le nombre de messages extraits à n = 2000, nous nous intéressons aux documents en langue française (lang).

```
#récupération des tweets
tweets <- searchTwitter("#presidentielle2017",n=2000,lang="fr")
```

Nous souhaitons prélever les messages relatifs au thème **#presidentielle2017** [*la fonction a été lancée le 02 mars 2017 à 14h45 (heure française)*]. Notons deux informations importantes :

- Nous ne savons pas comment Twitter choisit ces $n = 2000$ messages parmi l'ensemble des tweets possibles ;
- Selon le moment de la journée où l'on lance la commande, Twitter peut limiter le nombre de tweets retournés pour mieux gérer l'affluence sur le serveur vraisemblablement (à 10h du matin au mois de janvier, mes étudiants [24 machines] ont dû se contenter de 12 messages durant un de mes TD).

Dans le cas présent, nous avons bien récupéré les 2000 messages :

```
print(length(tweets))
```

```
## [1] 2000
```

Nous les explorons dans les sections suivantes.

3. Analyse des tweets

Une collection de tweets est une liste :

```
print(class(tweets))
```

```
## [1] "list"
```

Nous pouvons accéder à un message en utilisant un index. Par exemple, pour afficher le premier message, nous ferons :

```
print(tweets[[1]])
```

```
[1] "MdM_NPdC: RT @MdM_France: #Présidentielle2017 STOP aux idées reçues sur l'aide médicale pour les étrangers (#AME) ! @FrancoisFillon Vivez-vous #DansL"
```

3.1. Propriétés d'un tweet

Au-delà du texte brut, à chaque tweet est associé plusieurs propriétés.

- Le texte lui-même.

```
print(tweets[[1]]$text)
```

```
[1] "RT @MdM_France: #Présidentielle2017 STOP aux idées reçues sur l'aide médicale pour les étrangers (#AME) ! @FrancoisFillon Vivez-vous #DansL"
```

- L'auteur du texte (son **pseudo**). Il s'agit de "Médecins du Monde"" en l'occurrence (https://twitter.com/mdm_npdc).

```
print(tweets[[1]]$screenName)
```

```
## [1] "MdM_NPdC"
```

- L'identifiant du tweet

```
tweets[[1]]$id
```

```
## [1] "837297834185601025"
```

- Le pseudo de l'utilisateur auquel il a répondu.

```
tweets[[1]]$replyToSN
```

```
## character(0)
```

Il semble que ce tweet ne réponde pas à un interlocuteur direct.

- L'ID du message auquel il a répondu.

```
tweets[[1]]$replyToUID
```

```
## character(0)
```

Il ne s'agit pas d'une réponse à un autre tweet visiblement.

- Sa source. Le tweet a été envoyé à partir d'un iPad.

```
tweets[[1]]$statusSource
```

```
[1] "Twitter for iPad"
```

- La date de création.

```
tweets[[1]]$created
```

```
## [1] "2017-03-02 13:45:34 UTC"
```

Rappelons que j'ai effectué mon extraction à 14h45 heure française, ce qui correspond à 13h45 UTC. Le message était donc très récent lorsqu'il a été téléchargé.

- Est-ce que c'est un retweet ?

```
tweets[[1]]$isRetweet
```

```
## [1] TRUE
```

- Le message a été tronqué ?

```
tweets[[1]]$truncated
```

```
## [1] FALSE
```

- Le message a-t-il été mis en favori par d'autres personnes ?

```
tweets[[1]]$favorited
```

```
## [1] FALSE
```

- Le message a-t-il été retweeté ?

```
tweets[[1]]$retweeted
```

```
## [1] FALSE
```

- Le nombre de retweet le cas échéant ?

```
tweets[[1]]$retweetCount
```

```
## [1] 41
```

Très étrangement, le message n'a pas été retweeté et pourtant le nombre de retweet serait de 41. Comme c'est un message retweeté (voir $isRetweet), il doit s'agir du compteur global comptabilisant les retweets du message initial.

3.2. Stockage des tweets dans une structure data.frame

Puisque nous disposons d'une liste, il est aisé de la parcourir à l'aide d'une boucle ou des procédures *sapply()* ou *lapply()* pour accéder à l'ensemble des messages. Les traitements sont encore plus facilités lorsque nous passons par une structure data.frame via la commande **twListToDF()**. Nous disponsons ainsi d'un tableau rectangulaire avec n= 2000 lignes (tweets) et p = 16 colonnes.

```
#copier la liste dans une structure data.frame
df <- twListToDF(tweets)

#dimensions
print(dim(df))
```

```
## [1] 2000   16
```

Nous retrouvons les propriétés énumérées dans la section précédente.

```
print(colnames(df))
```

```
##  [1] "text"          "favorited"     "favoriteCount" "replyToSN"
##  [5] "created"       "truncated"     "replyToSID"    "id"
##  [9] "replyToUID"    "statusSource"  "screenName"    "retweetCount"
## [13] "isRetweet"     "retweeted"     "longitude"     "latitude"
```

Nous visualisons ci-après quelques propriétés des 5 premiers messages.

```
print(df[1:10,c('created','screenName','isRetweet','retweeted','retweetCount')])
```

```
##                 created     screenName isRetweet retweeted retweetCount
## 1  2017-03-02 13:45:34       MdM_NPdC      TRUE     FALSE           41
## 2  2017-03-02 13:45:32  RichardMolard     FALSE     FALSE            0
## 3  2017-03-02 13:45:29    Samsa_Asmas     FALSE     FALSE            0
## 4  2017-03-02 13:45:26    sireude1964      TRUE     FALSE            4
## 5  2017-03-02 13:45:26    RobertBadia      TRUE     FALSE           35
## 6  2017-03-02 13:45:25 PaulineColin19      TRUE     FALSE           15
## 7  2017-03-02 13:45:13   LeCorsaire22      TRUE     FALSE            2
## 8  2017-03-02 13:45:10      LaTribune     FALSE     FALSE            0
## 9  2017-03-02 13:45:10     DpgpSavoar     FALSE     FALSE            0
## 10 2017-03-02 13:45:08      Trader496     FALSE     FALSE            0
```

La structure peut-être être sauvegardée dans un fichier texte pour des traitements ultérieurs ou à l'aide d'autres outils de data mining.

```
#sauvegarde de la structure data.frame
write.table(df,"tweets.txt",sep="\t",quote=F)
```

3.3. Quelques statistiques descriptives

Nous retrouvons un format de tableau (data.frame) que nous savons très bien manipuler en data mining (mis à part la colonne $text que nous analyserons dans la partie « Text Mining »). Nous pouvons initier une première appréhension des données à l'aide de quelques statistiques descriptives simples.

Analyse des auteurs

Les auteurs de messages

Nous pouvons établir la liste des 10 auteurs les plus prolifiques avec le nombre de messages envoyés.

```
#comptage du nombre de message par auteurs
comptage <- table(df$screenName)
```

```
#tri décroissant
comptage <- sort(comptage,decreasing=TRUE)

#affichage des 10 premiers
print(comptage[1:10])

##
## Electionsde2017  FabriceBalique      i24NEWS_FR        f_minuit
##              18              12              11               9
##        LA_FEDOM    bmarilyne972   CollAssoUnies      Renaud1717
##               9               7               7               7
##   le63avecHAMON         Tian_A1
##               6               6
```

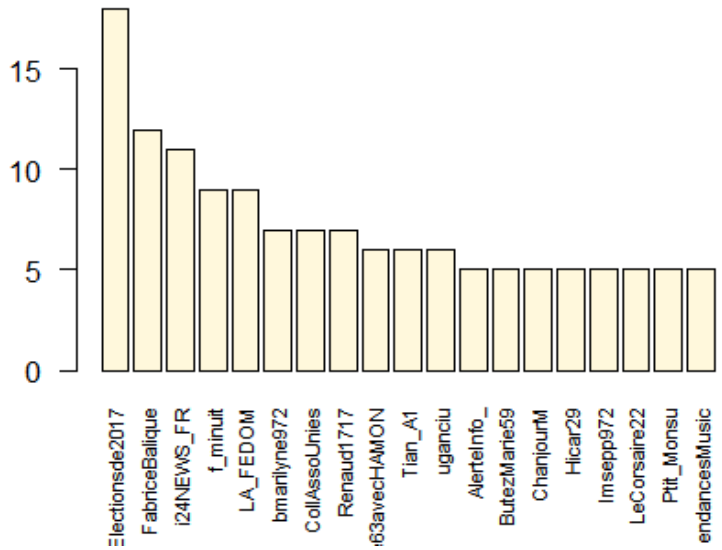
Finalement, la concentration des auteurs n'est pas très forte. Par rapport au nombre de messages ($n = 2000$), le nombre unique d'auteurs l'est également (1611).

```
print(length(unique(df$screenName)))

## [1] 1611
```

Nous pouvons représenter graphiquement la liste des auteurs ayant envoyé 5 messages ou plus.

```
#nous utilisons la variable comptage définie précédemment
barplot(comptage [comptage >= 5], las = 2,cex.names=0.7,col="cornsilk")
```



Les auteurs de messages originaux

Ce premier classement donne un premier point de vue sur l'activité des auteurs. Mais il peut être biaisé par le fait que certains messages sont en réalité de simples retweets. Pour identifier les auteurs de messages "originaux", qui amènent une réelle valeur ajoutée dans les échanges, nous essayons d'isoler les messages qui ne sont pas des retweets, puis nous comptabilisons de nouveau les auteurs.

Nous identifions tout d'abord les messages qui ne sont pas des retweets.

```
#liste des messages originaux
id_originaux <- which(!df$isRetweet)

#nombre de messages originaux
print(length(id_originaux))

## [1] 689
```

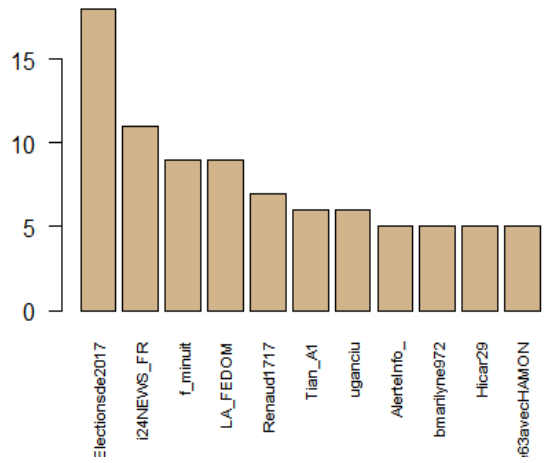
Ils sont au nombre 689 (sur les 2000 messages initiaux).

Nous réitérons l'opération de comptage.

```
#comptage du nombre de message par auteurs
comptage_bis <- table(df$screenName[id_originaux])

#tri décroissant
comptage_bis <- sort(comptage_bis,decreasing=TRUE)

#graphique de ceux qui ont plus de 5 (inclus) messages
barplot(comptage_bis [comptage_bis >= 5], las = 2,cex.names=0.7, col = "tan")
```



Les auteurs ayant rédigé 5 messages originaux ou plus sont plus rares (11), alors qu'ils étaient 19 si l'on comptabilisait les retweets. Certains, apparemment très actifs, ont disparu de la circulation.

Analyse de la popularité à travers les retweets

Les internautes retweetent les messages lorsqu'ils ont en apprécié la teneur. **Parmi les messages qui sont des retweets**, nous essayerons d'isoler les 2 messages qui sont les plus populaires.

```
#numéro des messages qui sont des retweets
idRetweets <- which(df$isRetweet)

#vecteur du compteur de retweet
#pour les messages retweetés
nombre_retweets <- df$retweetCount[idRetweets]

#index de tri décroissant selon le nombre
index <- order(nombre_retweets,decreasing=TRUE)

#2 premiers messages avec des auteurs et des identifiants différents
print(df[df$isRetweet,][index[1:2],c('screenName','id','retweetCount')])
```

```
##          screenName                  id retweetCount
## 121 DDiegodelaVerga 837294028982996992          678
## 415         meephon 837284795038195716          678
```

Nous avons 2 messages avec des auteurs et des identifiants distincts, qui sont répétés chacun 678 fois.

Qu'on ait deux fois la même valeur de répétition est assez intrigant. Après coup, on se rend qu'il s'agit du même message que les internautes aiment marteler visiblement.

```
#mais qui correspondent au même texte
print(df[df$isRetweet,][index[1:2],c('text')])
```

```
[1] "RT @fdeligne: À l"assassin! #elections #Presidentielle2017 #Fillon #dessin https://t.co/AdvyXAlPPo"
[2] "RT @fdeligne: À l"assassin! #elections #Presidentielle2017 #Fillon #dessin https://t.co/AdvyXAlPPo"
```

Ce résultat ne nous convient pas, il nous faut enlever les messages en doublon. Nous nous servons de la fonction **duplicated()** qui permet de les identifier.

```
#récupération du data.frame trié selon le nombre de retweets
#on ne travaille que sur les retweets (df$isRetweet)
dfRetweet <- df[df$isRetweet,][index,]

#première occurrence de chaque exemplaire de tweet
first <- !duplicated(dfRetweet$text)

#affichage des $2$ premiers éléments
print(dfRetweet$text[first][1:2])
```

[1] "RT @fdeligne: À l"assassin! #elections #Presidentielle2017 #Fillon #dessin https://t.co/AdvyXAlPPo"

[2] "RT @Conseil_constit: Décision 158 #PDR, le @Conseil_constit a validé 1717 #parrainages au 1er mars #Présidentielle2017 https://t.co/Qpb9Vnd"

Nous avons bien deux messages distincts maintenant, qui sont répétés respectivement 678 et 550 fois :

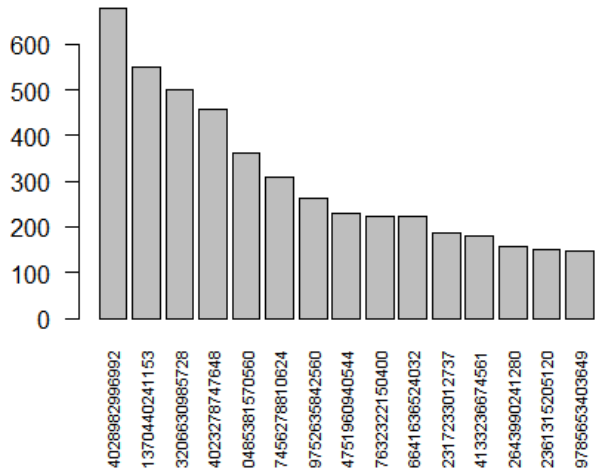
```
#affichage de leur nombre de répétition
print(dfRetweet$retweetCount[first][1:2])

## [1] 678 550
```

Nous pouvons représenter le nombre d'occurrence de 15 premiers tweets avec leur ID.

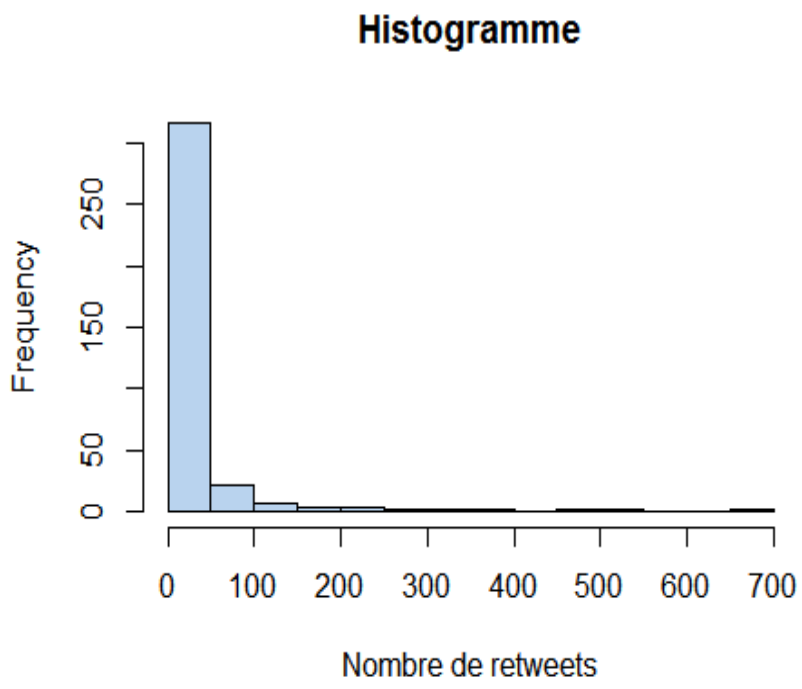
```
#data.frame correspondant aux premières occurrences
dfFirst <- dfRetweet[first,]

#graphique du nombre de retweets des messages les plus populaires
barplot(dfFirst$retweetCount[1:15], names.arg= dfFirst$id[1:15],las = 2,cex.names=0.7)
```



Ou encore afficher l'histogramme des fréquences du nombre de retweets.

```
hist(dfFirst$retweetCount,main="Histogramme",col="slategray2",xlab="Nombre de retweets")
```



Quelques tweets sont très populaires, un très grand nombre ont peu d'échos.

4. Analyse du contenu des tweets

4.1. Analyse des thèmes et des individus

Nous souhaitons dans un premier temps nous pencher sur les références aux thèmes (#) et aux auteurs (@) qui apparaissent dans les messages. Un premier nettoyage est nécessaire pour éliminer les éléments qui peuvent engendrer du bruit, gênant l'analyse.

Premier nettoyage des tweets

Elimination des doublons

Nous savons qu'il y a des répétitions dans les messages. Nous éliminons les doublons.

```
#data.frame avec les messages uniques
#les premières occurrences sont récupérées
dfUnique <- df[!duplicated(df$text),]
```

```
#nombre de tweets concernés
print(nrow(dfUnique))

## [1] 1045
```

Nous disposons de 1045 messages que nous collectons dans un vecteur spécifique :

```
#vecteur avec les messages
messages <- dfUnique$text

#taille du vecteur - vérification
print(length(messages))

## [1] 1045
```

Voici les messages n°8 :

```
print(messages[8])
```

[1] "#Presidentielle2017 : la justice entrave-t-elle la démocratie ? https://t.co/pP3xAQoj8j par @28minutes https://t.co/4Dnphg1bch"

Retrait des sauts de lignes et des références web

Plusieurs éléments que nous n'exploiterons pas peuvent perturber. Nous les supprimons et nous réaffichons le même message.

```
#retrait du saut de ligne \n
msgClean <- gsub("\n"," ",messages)

#retrait des URL
msgClean <- gsub('http\\S+\\s*',"",msgClean)

#retrait des espaces en trop
msgClean <- gsub("\\s+"," ",msgClean)

#retrait des "\"
msgClean <- gsub("[\\]","",msgClean)

#retrait des espaces en fin de texte
msgClean <- gsub("\\s*$","",msgClean)

#harmonisation de la casse - tout mettre en minuscule
msgClean <- tolower(msgClean)

#retrait des accents
msgClean <- gsub("[éèê]","e",msgClean)
msgClean <- gsub("[àâ]","a",msgClean)
msgClean <- gsub("[ùû]","u",msgClean)
```

```
#retrait de l'indicateur de retweet
msgClean <- gsub("rt ","",msgClean)

#vérification - affichage du document 8
print(msgClean[8])
```

```
[1] "#presidentielle2017 : la justice entrave-t-elle la democratie ? par @28minutes"
```

Remarque : Nous utilisons le mécanisme des pour retirer notamment les URL. La commande indique de remplacer (**gsub()**) par un caractère vide, le chaîne de caractères commençant par "http", suivie par une suite de longueur 1 ou plus de caractères qui ne sont pas des espaces "\\S+", et finissant un espace de longueur quelconque "\\s*".

En observant les messages, je me suis rendu compte certains ne différaient que par les URL qu'ils mettaient en référence. De nouveau donc, nous récupérons les exemplaires uniques de chaque tweet.

```
#enlever les doublons
msgClean <- msgClean[!duplicated(msgClean)]

#nombre de messages
print(length(msgClean))
```

```
## [1] 1019
```

Nous avons bien fait puisque nous sommes passés de 1045 à **1019 tweets**. Nous travaillerons à partir de ces documents dans ce qui suit.

Analyse des thèmes

Le caractère "#" joue un rôle particuier sur Twitter. Il permet de désigner un *hashtag*, un sujet relatif au message que l'on rédige ou en relation avec nos préoccupations. Plusieurs *hashtags* peuvent donc apparaître dans un même message. Nous recensons l'ensemble des thèmes cités sous forme de *hashtag* dans l'ensemble de nos tweets.

Tout d'abord, nous les collectons dans un vecteur. Ils sont au nombre de 2513 (pour $n = 2000$ messages).

```
#récupérer l'ensemble des mots délimtés par des ESPACE
all_mots <- unlist(strsplit(msgClean," "))

#un mot est un hashtag s'il débute par #
signature_hashtag <- regexpr("^#[[:alnum:]_]*",all_mots)
```

```
#récupérer l'ensemble des thèmes désignés par un "#" dans les messages
liste_hashtags <- regmatches(all_mots,signature_hashtag)

#nombre de hashtags recensés
print(length(liste_hashtags))
```

```
## [1] 2513
```

Puis, nous comptabilisons leur nombre d'apparition, en mettant en avant les 10 *hashtags* plus populaires.

```
#nombre d'apparition de chaque hashtag
nb_hashtags <- table(liste_hashtags)

#tri selon la fréquence décroissante
tri_nb_hashtags <- sort(nb_hashtags,decreasing=TRUE)

#affichage des 10 hastags les plus fréquents
print(tri_nb_hashtags[1:10])
```

```
## liste_hashtags
## #presidentielle2017            #fillon             #macron
##                 941                182                118
##        #programmeem         #fillongate         #fillon2017
##                  67                 52                 40
##           #enmarche       #penelopegate              #hamon
##                  38                 37                 23
##          #hamon2017
##                  22
```

#presidentielle2017 apparaît en tête de liste, c'est tout à fait normal puisque c'était le mot-clé utilisé pour extraire les tweets.

Nous affichons les thèmes sous forme de wordcloud en excluant #presidentielle2017 qui est par trop évident :

```
#chargement de la librairie
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
#affichage sauf #presidentielle 2017
wordcloud(names(tri_nb_hashtags)[-1],tri_nb_hashtags[-1],scale=c(3,.5),colors
=brewer.pal(6, "Dark2"))
```

Le second thème dominant est "fillon" qui apparaît sous différentes formes. Les messages ont été extraits le 02 mars 2017, rappelons-le.

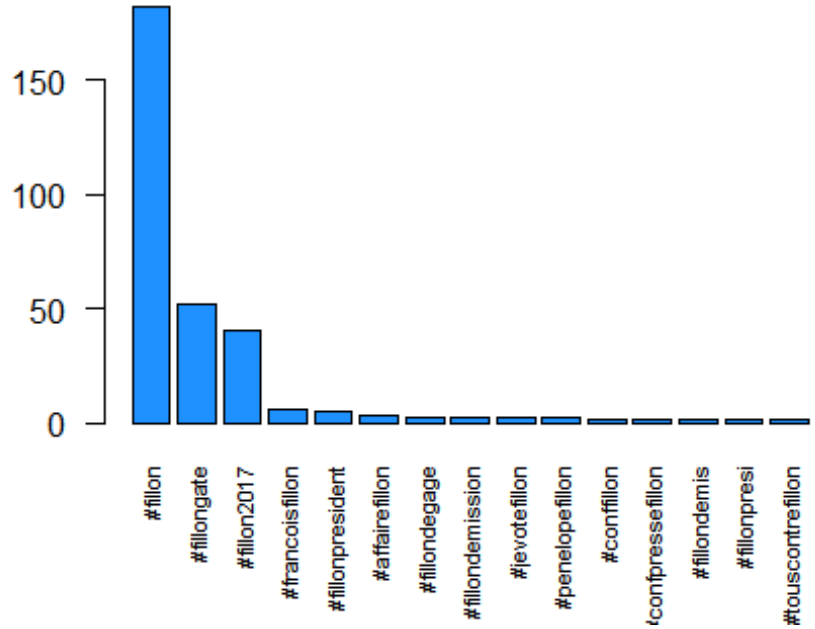
Nous pouvons identifier tous les hashtags associés au terme "fillon".

```
#liste des hashtags contenant fillon et leur nombre d'appartition
hashtags_fillon <- nb_hashtags[grep("fillon",names(nb_hashtags))]
print(sort(hashtags_fillon,decreasing=TRUE))
```

```
## liste_hashtags
##           #fillon       #fillongate       #fillon2017   #francoisfillon
##               182                52                40                 6
##  #fillonpresident    #affairefillon     #fillondegage  #fillondemission
##                 5                 3                 2                 2
##     #jevotefillon   #penelopefillon       #conffillon #confpressefillon
##                 2                 2                 1                 1
##      #fillondemis      #fillonpresi #touscontrefillon
##                 1                 1                 1
```

Que l'on peut mettre sous forme graphique :

```
barplot(sort(hashtags_fillon,decreasing=TRUE),las=2,cex.names=0.7,col="dodgerblue")
```



Analyse des individus via leur pseudo

Les individus sont identifiés par le caractère "@" (*screenname*). Ils apparaissent dans les messages parce qu'ils sont retweetés (ex. "RT @toto:"), ou parce qu'ils sont cités nommément. Dans ce qui suit, nous allons les comptabiliser.

Dans la liste des mots, nous repérons les termes commençant par "@" :

```
#un mot désigne un individu (pseudo) s'il débute par @
signature_individu <- regexpr("^@[[:alnum:]_]*",all_mots)

#récupérer l'ensemble des thèmes désignés par un "#" dans les messages
liste_individus <- regmatches(all_mots,signature_individu)

#nombre de hashtags recensés
print(length(liste_individus))

## [1] 868
```

Ici également, nous pouvons repérer les noms d'auteurs qui apparaissent le plus fréquemment dans les messages.

```
#nombre d'apparition des individus
nb_individus <- table(liste_individus)
```

```
#tri selon la fréquence décroissante
tri_nb_individus <- sort(nb_individus,decreasing=TRUE)

#affichage des 10 auteurs les plus fréquents
print(tri_nb_individus[1:10])
```

```
## liste_individus
##  @emmanuelmacron  @francoisfillon           @bfmtv     @benoithamon
##               56               42               14               13
##      @franceinfo  @fabricebalique        @la_fedom    @mlp_officiel
##               13               12               12               12
##        @f_minuit @conseil_constit
##               11                8
```

@emmanuelmacron et @francoisfillon sont les noms d'individus qui apparaissent le plus souvent, ce qui est somme toute cohérent avec les thèmes (*hashtags*) dégagés ci-dessus.

4.2. Text mining

Nous souhaitons dans cette section appliquer les méthodes de data mining aux tweets. Il faut pour cela que nous les transformions en matrice de données numériques. En effet, les algorithmes ne savent pas appréhender en l'état les documents textuels.

La représentation en sac de mots (*bag-of-words*) est la plus utilisée (Voir Text mining : la matrice documents termes - Diapos). Nous procédons en plusieurs étapes : nous repérons l'ensemble des termes distincts qui apparaissent dans les messages, ils vont constituer le dictionnaire ; nous créons alors une matrice où en ligne nous avons les tweets, en colonne les termes, et - dans sa version la plus fruste - à l'intersection ligne-colonne nous indiquons (1/0) la présence ou l'absence du terme dans le tweet. On parle de **matrice documents-termes**.

La construction du dictionnaire est fondamentale parce qu'il délimite les notions que nous prendrons en compte dans l'analyse textuelle.

Constitution et nettoyage du dictionnaire

Le dictionnaire est constitué à partir de la liste des mots. Plusieurs opérations de nettoyage sont nécessaires pour aboutir à une description pertinente. Pour ce faire, nous utiliserons les outils du package **tm** (https://cran.r-project.org/web/packages/tm/index.html).

Dans un premier temps, nous retirons des messages les pseudos et les hashtags parce qu'ils jouent un rôle spécifique. Nous souhaitons nous concentrer sur les termes qui donnent un sens aux messages.

```
#retrait des hashtags
msgCleanBis <- gsub("#[[:alnum:]_]*( |:|$)","",msgClean)

#retrait des pseudos
msgCleanBis <- gsub("@[[:alnum:]_]*( |:|$)","",msgCleanBis)

#vérification avec le document n°8
print(msgCleanBis[8])
```

```
## [1] ": la justice entrave-t-elle la democratie ? par "
```

Dans un second temps, nous transformons le vecteur de tweets en un format "Corpus" que savent manipuler les fonctions du package.

```
#importation de la libraire
library(tm)
```

```
## Loading required package: NLP
```

```
#transformation de la liste des tweets en un format interne
corpus <- Corpus(VectorSource(msgCleanBis))
print(corpus)
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 1019
```

Puis, nous enchaînons avec un second niveau de nettoyage.

```
#retrait des ponctuations
corpus <- tm_map(corpus,removePunctuation)

#retrait des nombres
corpus <- tm_map(corpus,removeNumbers)

#retrait des stopwords (mots outils)
corpus <- tm_map(corpus,removeWords,stopwords("french"))

#retirer les espaces en trop (s'il en reste encore)
corpus <- tm_map(corpus,stripWhitespace)

#vérification avec le document n°8
print(corpus[[8]]$content)
```

```
## [1] " justice entravetelle democratie "
```

Tout l'enjeu du nettoyage est là : évacuer ce qui peut constituer du bruit, tout en conservant l'information pertinente. Dans le cas présent, le sens est bien conservé. Il est question de la

justice qui entraverait la démocratie. La référence à l'émission de télé d'Arte "28 minutes" n'y est plus en revanche.

Création de la matrice documents-termes

Une fois le nettoyage effectué, nous pouvons créer la matrice documents-termes. Nous demandons une pondération binaire c.-à-d. seule la présence/absence des mots dans les tweets est recensée.

```
#création de la MDT à partir du corpus
mdt <- DocumentTermMatrix(corpus,control=list(weighting=weightBin))
print(mdt)
```

```
## <<DocumentTermMatrix (documents: 1019, terms: 2690)>>
## Non-/sparse entries: 5981/2735129
## Sparsity           : 100%
## Maximal term length: 108
## Weighting          : binary (bin)
```

Nous diposons d'un matrice avec 1045 tweets et 2692 termes.

Les termes les plus fréquents apparaissant au moins 60 fois, et par conséquent présents dans au moins 60 messages puisque nous avons adopté une pondération binaire, sont :

```
#termes apparaissant au moins 60 fois
print(findFreqTerms(mdt,60))
```

```
## [1] "cest"           "presidentielle" "programme"
```

Les résultats sont tout à fait cohérents avec la thématique "#presidentielle2017" étudiée.

L'objet "mdt" est dans un format *sparse*, pas toujours facile à manipuler. Nous le transformons en une matrice pleine.

```
m <- as.matrix(mdt)
print(dim(m))
```

```
## [1] 1019 2690
```

Ce qui nous permet de réaliser des opérations simples comme calculer la fréquence des mots et les trier. Nous affichons les 10 termes les plus fréquents.

```
#frequence des mots
freqMots <- colSums(m)

#tri par ordre décroissant
freqMots <- sort(freqMots,decreasing=TRUE)
```

```
#affichage des 10 mots les plus fréquents
print(freqMots[1:10])
```

```
## presidentielle      programme           cest         fillon       campagne
##             72             70             66             58             54
##      candidats       candidat         macron            via           plus
##             54             49             42             37             33
```

Nous constatons que le terme "candidat" se répète sous 2 formes, au singulier et au pluriel. Un nettoyage supplémentaire est nécessaire. Il faudrait revenir à la forme canonique des mots : on parle de lemmatisation. En anglais, ce n'est déjà pas évident, en français encore plus. Cet exemple montre bien à quel point le text mining est un sacerdoce. Au fur et à mesure que nous produisons des résultats, nous nous rendons compte que de nombreuses opérations de préparation de données manquent en amont... et il faut remettre le couvert.

Pour ce qui est des termes les plus rares, nous affichons le nombre de ceux qui apparaissent moins de 2 fois (inclus).

```
#termes n'apparaissant qu'une fois
print(length(which(freqMots<=2)))
```

```
## [1] 2200
```

Ils sont nombreux. Ils n'aident certainement pas à l'interprétation des tweets. Nous décidons de les éliminer.

```
#ne conserver que les termes apparaissant plus de 2 fois dans la matrice
mClean <- m[,colSums(m) > 2]
print(dim(mClean))
```

```
## [1] 1019  490
```

Il ne nous reste plus que 490 termes.

Un petit nuage de mots si cher aux data scientists pour visualiser tout cela. On s'en tient aux termes qui apparaissent au moins 10 fois pour éviter d'obtenir un fouillis.

```
#affichage sauf #presidentielle 2017
wordcloud(colnames(mClean),colSums(mClean),min.freq=10,scale=c(2,.5),colors=b
rewer.pal(6, "Dark2"))
```

Extraction des règles d'association

Nous disposons d'un tableau individus-variables, ultra-classique en data mining. Toutes les études sont possibles... une fois correctement établi le cahier de charges.

Il y a certainement beaucoup à faire à partir de ces données. Je vois sur le web de très nombreux exemples de traitements basés sur les analyses factorielles ou la classification automatique. Je me suis dit qu'il pouvait être intéressant d'explorer d'autres pistes. Je suis parti sur l'idée d'une extraction des itemsets fréquents. L'objectif est d'identifier les cooccurrences c.-à-d. les termes qui reviennent *souvent* conjointement dans les tweets.

Nous utiliserons le package **arules**.

```
#chargement de la librairie
library(arules)

## Loading required package: Matrix

##
## Attaching package: 'arules'

## The following object is masked from 'package:tm':
##
##     inspect
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write

#paramètres de l'extraction des itemsets
parametres <- list(supp=0.01,minlen=2, maxlen=3,target="frequent itemsets")

#extraction des itemsets
itemsets <- apriori(mClean,parameter=parametres)

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##          NA    0.1    1 none FALSE            TRUE       5    0.01      2
##  maxlen            target   ext
##       3 frequent itemsets FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 10
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[490 item(s), 1019 transaction(s)] done [0.00s].
## sorting and recoding items ... [63 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3

## Warning in apriori(mClean, parameter = parametres): Mining stopped (maxlen
## reached). Only patterns up to a length of 3 returned!

##  done [0.00s].
## writing ... [21 set(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

#affichage
inspect(itemsets)

##      items                       support
## [1]  {fillon,françois}           0.01177625
## [2]  {medicale,reçues}           0.01177625
## [3]  {idees,reçues}              0.01177625
## [4]  {etrangers,reçues}          0.01177625
## [5]  {reçues,stop}               0.01177625
## [6]  {idees,medicale}            0.01177625
## [7]  {etrangers,medicale}        0.01177625
## [8]  {medicale,stop}             0.01177625
## [9]  {etrangers,idees}           0.01177625
## [10] {idees,stop}                0.01177625
```

```
## [11] {etrangers,stop}            0.01177625
## [12] {idees,medicale,reçues}     0.01177625
## [13] {etrangers,medicale,reçues} 0.01177625
## [14] {medicale,reçues,stop}      0.01177625
## [15] {etrangers,idees,reçues}    0.01177625
## [16] {idees,reçues,stop}         0.01177625
## [17] {etrangers,reçues,stop}     0.01177625
## [18] {etrangers,idees,medicale}  0.01177625
## [19] {idees,medicale,stop}       0.01177625
## [20] {etrangers,medicale,stop}   0.01177625
## [21] {etrangers,idees,stop}      0.01177625
```

Nous comprenons parfaitement l'association entre "François" et "Fillon".

Pour les autres itemsets, nous identifions le message en cause : "Stop aux idées reçues de certains candidats...". Il a été maintes fois retweeté par les intervenants. Normalement, ils sont en doublons, ils auraient dû être éliminés. En regardant de près la base initiale, je me suis rendu compte qu'ils ont été plus ou moins tronqués selon le pseudonyme du retweeteur, faussant ainsi les comparaisons.

Argh ! De nouveau, il faut remettre l'ouvrage sur le métier et revenir au nettoyage des données... Je m'arrêterai là en ce qui me concerne, mais cet exemple illustre parfaitement combien il est difficile de travailler sur des bases réelles ! Encore plus quand il s'agit de bases textuelles !

5. Conclusion

L'étude des tweets est un axe fort de l'analyse des réseaux sociaux parce que Twitter est devenu un vecteur de communication important. Cet exemple montre qu'il est aisé d'initier une première analyse à partir de données extraites directement en ligne. Lorsqu'il s'agit d'aller dans le détail, explorer en profondeur les informations que recèlent les messages, l'affaire est tout autre. La phase de préparation des données prend une importance comme jamais. De la rigueur dont nous faisons preuve dans cette étape dépendra la crédibilité des résultats que nous produirons.