Climate and Data

Session 5 - Extracting and analyzing textual data using R

Jean-Baptiste Guiffard

November 29, 2024

La manipulation des données textuelles sur R

La production et l'analyse de données qualitatives

- De plus en plus important de maîtriser des méthodes d'analyse de données qualitatives (et particulièrement textuelles).
- Les packages et fonctions Rfacilitent le processus d'identification, de manipulation et d'analyse des textes.

Que pouvons-nous faire ?

- Analyse de la fréquence des mots
- Comparaison des textes
- Sentiment Analysis
- Des nuages de mots
- Des réseaux de co-occurence
- Analyse des thématiques et leur évolution en fonction du temps

Quelques références

- https://m-clark.github.io/text-analysis-with-R/string-theory.html#basic-text-functionality
- https://www.red-gate.com/simple-talk/databases/sql-server/bi-sql-server/text-mining-and-sentiment-analysis-with-r/
- https://www.tidytextmining.com/topicmodeling.html

Partons d'un exemple simple...

```
"L'influence humaine a réchauffé l'atmosphère, l'océan et les terres."
```

```
## [1] "L'influence humaine a réchauffé l'atmosphère, l'océan et les terres."
```

Nous pouvons en faire un objet sur R...

```
ma_phrase <- "L'influence humaine a réchauffé l'atmosphère, l'océan et les terres."
```

Nous pouvons avoir un vecteur de *characters* (des chaînes de caracters séparées par des virgules) qui peut aussi être utilisé dans le cadre d'une variable dans une *data.frame*.

Vérifier que le scalaire, le vecteur ou la variable étudié est constitué d'une ou de chaînes de caractères.

```
is.character(ma_phrase)
```

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

```
nchar(ma_phrase) #nombre de caractères dans le string
```

```
## [1] 68
```

Le package **stringr** (qui se charge aussi avec le package **tidyverse**) fournit un ensemble cohérent de fonctions conçues pour rendre le travail avec les chaînes de caractères aussi facile que possible.

Quelques fonctions basiques (II)

Détecter un champ dans une chaîne de caractères...

```
#install.packages("stringr")
library(stringr)
str_detect(ma_phrase,"influence")
```

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

```
str_detect(ma_phrase,'calamar')
```

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

Remplacer une partie d'une chaîne de caractères...

```
ma_phrase <- str_replace(ma_phrase,'humaine', "de l'homme")
```

Quelques fonctions basiques (III)

Remplacer plusieurs éléments d'un seul coup (gsub en séparant les éléments avec |)... Nous allons l'utiliser régulièrement pour le nettoyage des variables "textuelles".

```
ma_phrase <- gsub("'|,", " ", ma_phrase)
print(ma_phrase)
```

```
## [1] "L influence de l homme a réchauffé l atmosphère  l océan et les terres."
```

Nos premières analyses de corpus

Le package tidytext

```
# install.packages('tidytext')
library(tidytext)
```

- Un package efficace d'analyse textuelle (qui associe des fonctionnalités déjà présentes dans plusieurs autres packages : dplyr, broom, tidyr and ggplot2).
- Le format tidy text est un tableau avec un token par ligne.
- Un token est un mot, ou un phrase ou un n-gram.

Le vocabulaire

- Un **string** (ou un texte ou un document) : peut être stocké sous forme de chaîne de caractères, ou bien sous forme d'un vecteur de chaînes de caractères.
- Un **corpus** (ou une collection) : C'est un objet qui contient des strings (ou textes) avec des détails et métadonnées supplémentaires.
- La **matrice Document-term** : C'est une matrice décrivant un corpus de documents avec une ligne pour chaque document et une colonne pour chaque terme. Les valeurs à l'intérieur de la matrice sont soit un comptage de mots ou bien tf-idf.

Créer le corpus

```r
#install.packages("tidyft") #Pour le recodage des variables textuelles
library(tidyft)
bdd_speech <- read.csv2('full_bdd_speech_2024.csv') %>%
  as.data.table() %>%
  utf8_encoding(titles) %>%
  utf8_encoding(dates) %>%
  as.data.frame()
#head(bdd_speech$titles, n=6)
```

Nous pouvons utiliser la fonction gsub pour nettoyer des variables “textuelles”:

```
bdd_speech$speech <- gsub("[[:punct:]]", " ", bdd_speech$speech)
bdd_speech$speech <- gsub('[[:digit:]]+', '', bdd_speech$speech)
bdd_speech$speech <- gsub("[\r\n]", "", bdd_speech$speech)
#head(bdd_speech$speech, n=4)
```

La fonction unnest() extrait les tokens, ou mots particuliers d'un ensemble de données, de la colonne "texte" et les distribue dans des lignes individuelles avec les métadonnées correspondantes.

```
tidy_text <- tibble(bdd_speech) %>%
  unnest_tokens("word", speech)

head(tidy_text, n=4)
```

```
## # A tibble: 4 x 6
##       X dates      titles                                    category links word
##   <int> <chr>      <chr>                                     <chr>    <chr> <chr>
## 1     1 15/07/2024 Remise du rapport sur la fiscalité loca~ conomie  /rap~ remi~
## 2     1 15/07/2024 Remise du rapport sur la fiscalité loca~ conomie  /rap~ du
## 3     1 15/07/2024 Remise du rapport sur la fiscalité loca~ conomie  /rap~ rapp~
## 4     1 15/07/2024 Remise du rapport sur la fiscalité loca~ conomie  /rap~ sur
```

Retirer les stopwords

Les **stopwords** → sont un ensemble de mots couramment utilisés dans une langue. Lorsqu'on traite le langage naturel, nous voulons filtrer ces mots de nos données.

- Chargement des stopwords français
- On peut utiliser anti_join() pour trouver les stop words qui apparaissent parmi nos tokens et les supprimer.

```
#install.packages("lsa") # packages pour le chargement de stopwords (pour différentes lang
library(lsa)
data(stopwords_fr)
df_stopwords_fr <- data.frame(word=stopwords_fr,
                              lexicon = "?")
tidy_text <- tidy_text %>% dplyr::anti_join(df_stopwords_fr)
head(tidy_text, n=4)
```

```
## # A tibble: 4 x 6
##       X dates      titles                                    category links word
##   <int> <chr>      <chr>                                     <chr>    <chr> <chr>
## 1     1 15/07/2024 Remise du rapport sur la fiscalité loca~ conomie  /rap~ remi~
## 2     1 15/07/2024 Remise du rapport sur la fiscalité loca~ conomie  /rap~ rapp~
```

Préparer l'analyse de l'occurence des mots

Avec cette nouvelle base de données, nous pouvons construire une matrice qui contient d'une part, les mots qui apparaissent dans le corpus et d'autre part, leur fréquence d'apparition.

```
head(tidy_text %>% dplyr::count(word, sort = TRUE))
```

```
## # A tibble: 6 x 2
##   word          n
##   <chr>     <int>
## 1 france     3839
## 2 ministre   2841
## 3 président  2492
## 4 messieurs  2343
## 5 mesdames   2276
## 6 pays       2208
```

Compléter la liste des stopwords (I)

Des mots spécifiques à notre corpus peuvent revenir de manière répétées sans être forcément très informatif.

```
new_stop_words_fr <- data.frame("word" = c("mme", "janvier", "février","mars","avril","mai"
tidy_text <- tidy_text %>% dplyr::anti_join(new_stop_words_fr)
```

```
## Joining with `by = join_by(word)`
```

```
head(tidy_text %>% dplyr::count(word, sort = FALSE))
```

```
## # A tibble: 6 x 2
##   word          n
##   <chr>     <int>
## 1 aadopté       1
## 2 aah           3
## 3 aannoncé      1
## 4 aarhus        1
## 5 aaugmenté     1
## 6 ab            2
```

Compléter la liste des stopwords (II)

```
head(tidy_text %>% dplyr::count(word, sort = TRUE))
```

```
## # A tibble: 6 x 2
##   word          n
##   <chr>     <int>
## 1 france     3839
## 2 ministre   2841
## 3 président  2492
## 4 messieurs  2343
## 5 mesdames   2276
## 6 pays       2208
```

Retrouver la racine de chaque mot

Les tokens doivent être "stemmed" → réduction des mots à leur racine, leur base ou leur forme.

```
tidy_text_stems <- tidy_text %>%
  mutate_at("word", funs(wordStem((.), language="fr")))

#head(tidy_text_stems)
```

Notre premier nuage de mots

```
#install.packages("wordcloud")
library(wordcloud)

tidy_text %>%
  dplyr::count(word) %>%
  with(wordcloud(word, n, min.freq = 1000, colors = brewer.pal(8, "Dark2")))
```

Améliorer les enseignements d'un nuage de mots


```
new_stop_words_fr2 <- data.frame("word" = c("affaires","ministre", "ministres","communiqué

tidy_text <- tidy_text %>% dplyr::anti_join(new_stop_words_fr2)
```

```
## Joining with `by = join_by(word)`
```

```
tidy_text_stems <- tidy_text %>%
  mutate_at("word", funs(wordStem((.), language="fr")))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```


nouveau
collect
franc
devon
présent
national
an
temp
pris
annéé
public
discour
messieur
parl
not
aid
rest
cas
sant
servic
permettr
territoir
social
port
euros
évident
prochain
plan
seul
état
entrepris
invest
engag
commun
semain
local
besoin
économ
réform
fac
fin
sujet
chos
disposit
respons
pouvoir
part
polit
mis
droit
plac
général
action
transform
compt
moyen
égal
concitoyen
loi
vi
object
don
mesur
région
mair
financ
accompagn
organis
situat
fois
sécur
jour
travail
nouvel
publiqu
mond
nombr
forc
grand
demand
cris
élus
sais
gouvern
mois
européen
etat
import
développ
président
pay
républ
mesdam
sort
pens
derni
projet
monsieur
rappel
question

TF-IDF (*term frequency-inverse document frequency*)

- Mesure statistique → Importance d'un terme contenu dans un texte vis-à-vis d'un corpus de texte. Le poids du mot varie en fonction du nombre d'occurences du mot dans le texte et en fonction du nombre de la fréquence du mot dans le corpus de documents.

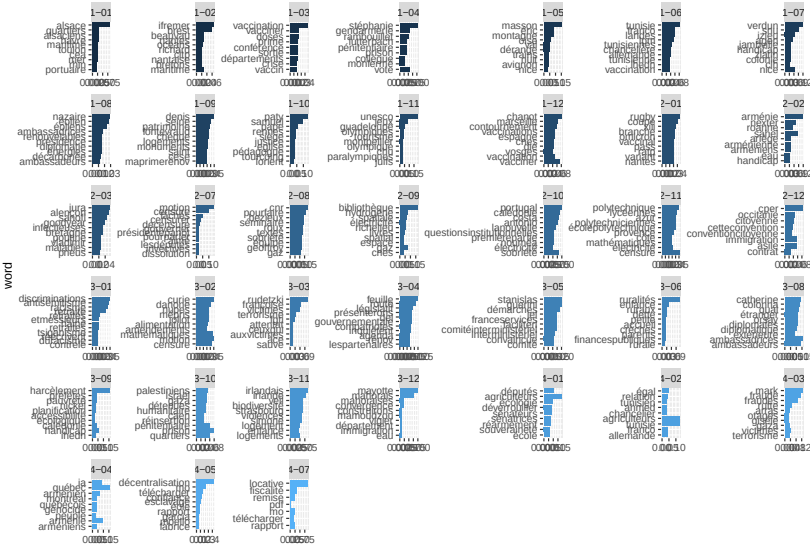
```
tidy_text_tfidf <- tidy_text %>%
  dplyr::count(word, dates) %>%
  bind_tf_idf(word, dates, n) %>%
  dplyr::arrange(desc(tf_idf))

head(tidy_text_tfidf)
```

```
## # A tibble: 6 x 6
##   word             dates          n    tf   idf tf_idf
##   <chr>            <chr>      <int> <dbl> <dbl>  <dbl>
## 1 locative         15/07/2024     2 0.182  4.97  0.904
## 2 debre            03/11/2020     2 0.111  5.38  0.598
## 3 décentralisation 31/05/2024     2 0.2    2.22  0.445
## 4 fiscalité        15/07/2024     2 0.182  1.90  0.345
## 5 lettre           03/11/2020     2 0.111  3.08  0.342
```

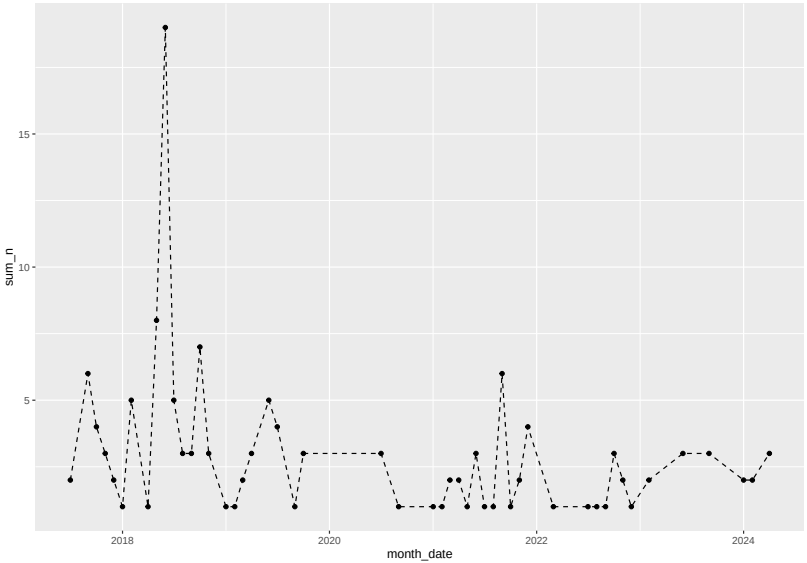
La matrice TF-IDF

```
tidy_text_tfidf$dates <- as.Date(tidy_text_tfidf$dates, format = c("%d/%m/%Y"))
tidy_text_tfidf$month <- paste("01/",format(tidy_text_tfidf$dates, "%m"),"/",format(tidy_te
tidy_text_tfidf$month_date <- as.Date(tidy_text_tfidf$month, format=c('%d/%m/%Y'))
```

Evolution des mots surprésentés par période



L'occurence des mots de l'écologie au travers du temps



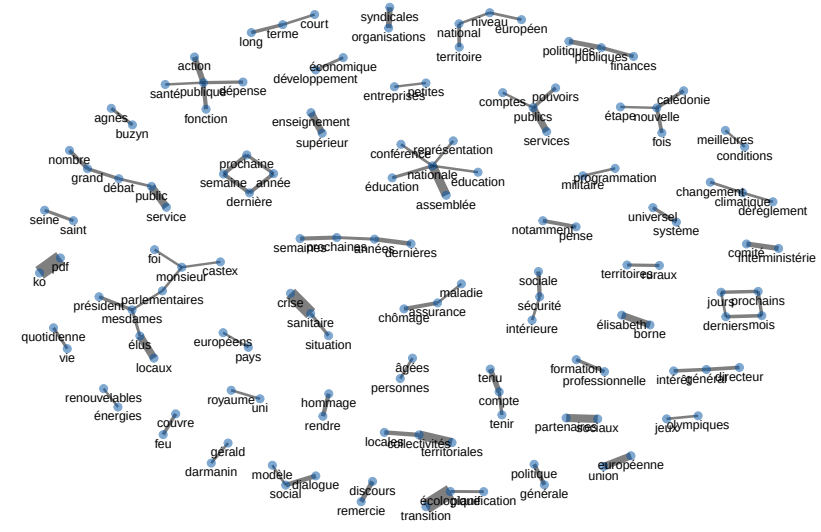
L'association entre les mots

cf other presentation

Un graph d'association des mots

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

Réseau des mots − (2007−2023)



Exercice : Analyse des rapports du GIEC

Extraire de l'information textuelle depuis un pdf


```
#install.packages("pdftools")
library("pdftools")


pdf.text_report_2001 <- pdftools::pdf_text("seance_5/2001_policy_report.pdf")
pdf.text_report_2007 <- pdftools::pdf_text("seance_5/2007_policy_report.pdf")
pdf.text_report_2014 <- pdftools::pdf_text("seance_5/2014_policy_report.pdf")

# selection d'une page en particulier
# cat(pdf.text_report_2001[[8]])

pdf.text_report_2001<-unlist(pdf.text_report_2001)
pdf.text_report_2001<-tolower(pdf.text_report_2001)
```