Extraire le contenu d'un pdf avec R

Introduction

R nous permet d'extraire le contenu de divers types de fichiers, y compris les fichiers PDF. Cependant, nous importerons tout le contenu de la page, ce qui n'est pas toujours le comportement souhaité car nous ne sommes intéressés que par une partie (ou des parties spécifiques) du document.

Dans ce travail on va :

• Extraire le contenu d'un fichier PDF en R (deux packages)

• Nettoyer le résultat afin de pouvoir lancer des analyses sémantiques

1. Extraire le contenu d'un fichier PDF en R (deux techniques)

1.1 Package pdftools

Les articles scientifiques sont généralement verrouillés au format PDF, un format conçu principalement pour l'impression, mais pas si idéal pour la recherche ou l'indexation. Le nouveau package pdftools permet d'extraire du texte et des métadonnées à partir de fichiers pdf dans R. À partir du texte brut extrait, on peut trouver des articles traitant d'un médicament ou d'un nom d'espèce particulier, sans avoir à compter sur des éditeurs fournissant des métadonnées ou des moteurs de recherche payants.

Les pdftools chevauchent légèrement le package Rpoppler de Kurt Hornik. La principale motivation derrière le développement de pdftools était que Rpoppler dépend de glib, qui ne fonctionne pas bien sur Mac et Windows. Le package pdftools utilise l'interface poppler c ++ avec Rcpp, ce qui se traduit par une implémentation plus légère et plus portable.(STHDA, n.d.)

Alors notre première technique consiste à l'utilisation du package pdftools disponible sur le CRAN:

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

Après l'installation de package on va importer le contenu du pdf, pour faire cela on va utiliser La fonction pdf_text qui va directement importer le text brut sous la forme d'un vecteur de type character avec des espaces pour représenter l'espace vide et des pour les sauts de ligne.

Puis on va utiliser strsplit pour séparer les lignes les unes des autres parce que ça serait un peu compliqué et non pratique d'avoir toute la page dans un seul élément.

On utilise la fonction cat qui affiche de façon simple les résultats sous forme de texte dans la console et permet l'export des résultats dans un objet

```
#install.packages(pdftools)
library(pdftools)
download.file("https://www.btboces.org/Downloads/I%20Have%20a%20Dream%20by%20Martin%20Luther%20King%20Jr.pdf","I%
20Have%20a%20Dream%20by%20Martin%20Luther%20King%20Jr.pdf", mode = "wb")
text <- pdf_text("I%20Have%20a%20Dream%20by%20Martin%20Luther%20King%20Jr.pdf")
text1 <- strsplit(text, "\n")
cat(text[1])
```

1.2 Le package tm

tm est le paquet "text mining" le plus populaire de R. Comme on va surtout travailler avec ce paquet, il va falloir l'installer si nécessaire.

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

Le paquet tm est conçu pour marcher avec une variété de formats: textes simples, articles/papiers en PDF ou Word, documents Web (HTML, XML, SGML), etc. Il fournit entre autre les fonctionnalités suivantes:

• Un dispositif pour analyser des corpus (une structure de données avec des fonctions de construction)

• Une fonction pour appliquer des fonctions à l'ensemble des textes d'un corpus.

• Des fonctions nouvelles pour l'analyse de textes("Tutoriel Tm Text Mining Package," n.d.)

a. Importation de documents et corpus

Pour bien simplifier ce travail on va l'appliquer sur un exemple. Tout d'abord on a 2 documents PDF stockés dans un « directory » sous le nom de docs. Les deux documents PDF sont :

Un speech de martin Luther king : https://www.btboces.org/Downloads/I%20Have%20a%20Dream%20by%20Martin%20Luther%20King%20Jr.pdf

Un article sur la liberté d'expression : http://www.supremecourt.ge/files/upload-file/pdf/article10eng.pdf

L'idée est de faire une certaine analyse et comparaison entre ces deux documents. Afin d'importer ces deux documents et extraire leurs contenus, on va créer une collection de documents stockés dans la structure R « corpus ».

On doit indiquer la source du corpus (où le trouver) et la méthode pour lire les divers fichiers (ou autre sources)

```
docs <- getwd()
my_corpus <- VCorpus(DirSource(docs, pattern = ".pdf"), readerControl = list(reader = readPDF))
```

Pour vérifier les contenus de ces documents on utilise la fonction « inspect » :

```
inspect(my_corpus)
writeLines(as.character(my_corpus[[1]]))
```

b. Nettoyage du contenu

Après avoir importé ces documents, on doit nettoyer les données et les contenus extraits.

Le nettoyage des contenus ne se fait pas toujours de la même façon, il dépend de l'objectif de l'analyse.

En ce qui concerne notre exemple, la première étape sera la suppression des ponctuations, cependant on doit s'assurer que il y a un espace entres ces ponctuations et le contenu du document afin de protéger les données. On va se baser sur la fonction content_transformer pour créer une fonction toSpace qui va nous permettre de mettre un espace entre les ponctuations et le contenu du PDF.(**???**)

```
toSpace<-content_transformer(function(x,pattern) {return(gsub(pattern," ",x))})
my_corpus<-tm_map(my_corpus,toSpace,"-")
my_corpus<-tm_map(my_corpus,toSpace,",")
my_corpus<-tm_map(my_corpus,toSpace,":")
my_corpus<-tm_map(my_corpus,toSpace,"--")
my_corpus<-tm_map(my_corpus,toSpace,"'")
```

Après ces changements on peut procéder à la suppression des ponctuations en utilisant la fonction « removePunctuation »

```
my_corpus<-tm_map(my_corpus, removePunctuation)
```

Comme R est un langage qui est sensible à la casse, on va rendre toutes les lettres dans les textes en minuscule avec la fonction « tolower ».

```
my_corpus<- tm_map(my_corpus, content_transformer(tolower))
```

Ensuite on va supprimer les nombres, cependant dans certains cas on aura besoin des nombres donc il faut faire le nettoyage des contenus avec une précision.

Comme on aura pas besoin des nombres dans notre exemple donc on va éliminer les nombres avec la fonction « removeNumbers ».

```
my_corpus<- tm_map(my_corpus, removeNumbers)
```

L'étape suivante est de supprimer les mots vides comme: and , or, if , yet ....

Pour faire cela on va utiliser la fonction « removewords » et « stopwords » en précisant la langue anglaise puisque ces deux documents sont en anglais.

```
my_corpus<- tm_map(my_corpus, removeWords, stopwords("english"))
```

Puis on procède à la suppression des espaces extrèmes avec la fonction « stripWhitespace » .

```
my_corpus<- tm_map(my_corpus, stripWhitespace)
```

Ensuite on va procéder au « stemming », autrement dit la désuffixation du contenu afin d'avoir que les racines des mots, car dans le processus de l'analyse des textes on s'intéresse pas au format des mots mais plutôt on s'intéresse à faire une analyse précise et fiable.

Le package qui va nous permettre à effectuer le « stemming » est le package « SnowballC », en effet Snowballc est une interface R vers la bibliothèque C 'libstemmer' qui implémente L'algorithme "The Porter stemming" pour regrouper les mots en un root pour faciliter la comparaison du vocabulaire.

Les langues actuellement prises en charge sont : Danois, néerlandais, anglais, finnois, français, allemand, hongrois, italien, Norvégien, portugais, roumain, russe, espagnol, suédois et turc.

```
#install.packages(Snowballc)
library(Snowballc)
```

on aura aussi besoin de la fonction « stemDocument» qui effectuer le stemming des mots

```
my_corpus<- tm_map(my_corpus, stemDocument)
```

La dernière étape de nettoyage est la création d'une matrice documents-termes (Angl: Document Term Matrix (DTM))qui liste la fréquence de mots par document. Il existe deux variantes, un matrice "documents par termes" ou une matrice "termes par documents".

Pour construire une matrice il faut utiliser « DocumentTermMatrix »

```
dtm <- DocumentTermMatrix(my_corpus)
inspect(dtm)
```

On aura comme résultat une matrice qui résume les nombres des mots par document.

Après avoir crée cette matrice on passe à l'étape de l'analyse .

2. **Analyse des textes**

Le text mining regroupe l'ensemble des techniques de data management et de data mining permettant le traitement des données particulières que sont les données textuelles. Par données textuelles, on entend par exemple les corpus de textes, les réponses aux questions ouvertes d'un questionnaire, les champs texte d'une application métier où des conseillers clientèle saisissent en temps réel les informations que leur donnent les clients, les mails, les posts sur les réseaux sociaux, les articles, les rapports ...

Un des aspects centraux du text mining est de transformer ces données textuelles peu structurées – si ce n'est par la langue utilisée – en données exploitables par les algorithmes classiques de data mining. Il s'agit tout simplement de transformer un texte brut en tableau de données indispensable aux analystes chargés d'en dégager du sens. Il s'agit ensuite de déployer les méthodes statistiques les plus à même de répondre à une problématique donnée.(LUCAS, n.d.)

Après avoir créé cette matrice, on peut passer à l'étape de l'analyse en utilisant des techniques quantitatives .

En effet,pour savoir la fréquence de chaque mot dans le corpus on utilise la fonction « colSums ».

```
freq<-colSums(as.matrix(dtm))
```

On peut aussi avoir l'ordre décroissant des fréquences des mots utilisés dans le contenu de corpus avec la fonction « order ».

```
ord<-order(freq,decreasing = TRUE)
```

Puis on peut afficher les mots les plus utilisés avec la fonction « head ».

```
freq[tail(ord)]
```

Ou afficher les mots les moins utilisés avec la fonction « tail »

```
freq[head(ord)]
```

2.1 *Le Wordcloud*

Les nuages de mots sont des outils de visualisation . Ils présentent des données textuelles dans un format simple et clair, celui d'un nuage dans lequel la taille des mots dépend de leurs fréquences respectives.

Les nuages de mots sont d'excellents outils de communication. Ils sont extrêmement pratiques pour tous ceux qui souhaitent communiquer des informations de base basées sur des données textuelles - que ce soit pour analyser un discours, capturer la conversation sur les réseaux sociaux ou rendre compte des avis des clients.

les nuages de mots nous permettent de tirer rapidement plusieurs aperçus, ce qui permet une certaine flexibilité dans leur interprétation. Leur format visuel nous stimule à réfléchir et à tirer le meilleur aperçu en fonction de ce que nous souhaitons analyser.

Pour générer des nuages de mots on doit télécharger le package wordcloud dans R ainsi que le package RcolorBrewer pour les couleurs.(Rui, n.d.)

On va appliquer le wordcloud sur le speech de Martin Luther King :

a. téléchargement des packages

Pour générer des nuages de mots, vous devez télécharger le package wordcloud dans R ainsi que le package RcolorBrewer pour les couleurs

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

#install.packages(RColorBrewer)
library(RColorBrewer)
```

b. Création d'une matrice de termes de document*

On va créer un nouveau corpus où on mets notre document

Par la suit on crée un dataframe contenant chaque mot dans la première colonne et leur fréquence dans la deuxième colonne.

Cela peut être fait avec la matrice de termes de document avec la fonction TermDocumentMatrix du package tm.

```
article <- Corpus(VectorSource(text))
tm <- TermDocumentMatrix(article)
matrix <- as.matrix(tm)
words <- sort(rowSums(matrix),decreasing=TRUE)
df <- data.frame(word = names(words),freq=words)
```

c. Générer le nuage de mots

Le package wordcloud est le moyen le plus classique de générer un nuage de mots. La ligne de code suivante vous montre comment définir correctement les arguments. A titre d'exemple, j'ai choisi de travailler avec les documents précedents qui parlent de la liberté .

```
set.seed(1234)
wordcloud(words = df$word, freq = df$freq, min.freq = 1, max.words=200, random.order=FALSE, rot.per=0.35,  colors
=brewer.pal(8, "Dark2"))
```

the
and
freedom
that
will
our
not
we
let
us
free
let freedom
ring
from
every
mountain
side
let freedom ring
when
this
day
let freedom ring from every mountain side
let freedom ring from every mountain side when this day
let freedom ring from every mountain side when this day we
let freedom ring from every mountain side when this day we let
let freedom ring from every mountain side when this day we let us
let freedom ring from every mountain side when this day we let us all
let freedom ring from every mountain side when this day we let us all join
let freedom ring from every mountain side when this day we let us all join hand
let freedom ring from every mountain side when this day we let us all join hand
let freedom ring from every mountain side when this day we let us all join hand