

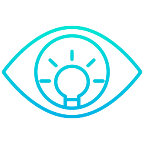
**Text Mining**

**Société :** INRAE

**Référence du document :** SE2011\_FormationINRAE

**Date :** 12/11/2020

**Version :** V1



L’autre regard

sur la **data**

**Seenovate**

8 rue de la Bourse

69002 LYON

**Contact**

Frank Sauvage

frank.sauvage@seenovate.com

Téléphone 06 64 03 13 91

**VERSIONS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Collaborateur** | **Modifications apportées** |
| 12/11/2020 | V1 | FSA |  |
| 29/09/2022 | V1.1 | VLE | Ajout de références + correction bugs |

**SOMMAIRE**

[I. Introduction 5](#_Toc57274627)

[A. Contexte et champs d’application 5](#_Toc57274628)

[B. principes généraux 5](#_Toc57274629)

[1. Degré de structuration des données 5](#_Toc57274630)

[2. Notion de corpus, de document et d’unité de sens 7](#_Toc57274631)

[II. Préparation des données 8](#_Toc57274632)

[A. Formats de données textuelles 8](#_Toc57274633)

[B. Préparation dans R 9](#_Toc57274634)

[1. Packages et workflow 9](#_Toc57274635)

[2. Tokenization 9](#_Toc57274636)

[3. Suppression des mots non informatifs ou « stop words » 11](#_Toc57274637)

[4. Lemmatisation 11](#_Toc57274638)

[III. Analyse descriptive 13](#_Toc57274639)

[A. Fréquence des mots 13](#_Toc57274640)

[B. Statistique tf-idf 16](#_Toc57274641)

[1. Définition 16](#_Toc57274642)

[2. Illustration 16](#_Toc57274643)

[IV. Analyse de sentiment (Sentiment analysis) 22](#_Toc57274644)

[A. Les dictionnaires de sentiment 22](#_Toc57274645)

[B. Le sentiment à travers un corpus de texte 24](#_Toc57274646)

[C. L’analyse des contribution au sentiment 27](#_Toc57274647)

[1. Mots positifs et négatifs les plus fréquents 27](#_Toc57274648)

[2. Représentations graphiques 28](#_Toc57274649)

[D. Bilan 29](#_Toc57274650)

[A. Complément sur la lemmatisation et le sentiment 29](#_Toc57274651)

[V. Modélisation de la thématique 32](#_Toc57274652)

[A. Contexte et workflow 32](#_Toc57274653)

[B. Approche LDA 32](#_Toc57274654)

[C. Ajustement de la LDA 33](#_Toc57274655)

[D. Obtention des probabilités "par-mot-par-thème" 33](#_Toc57274656)

[E. Interprétation des thèmes 34](#_Toc57274657)

[F. Compléments à l'interprétation 34](#_Toc57274658)

[1. Différences principales entre les thèmes 34](#_Toc57274659)

[2. Proportions de chaque thème dans les documents 36](#_Toc57274660)

[G. Exploration de l'impact de k sur la distribution des gammas par document 38](#_Toc57274661)

[VI. Approche supervisée 39](#_Toc57274662)

[A. Préparation des données 39](#_Toc57274663)

[1. Stemming des termes de AssociatedPress 39](#_Toc57274664)

[2. Reconstruction de la matrice Document-Term 40](#_Toc57274665)

[B. Identification et allocation des documents à différents thèmes 40](#_Toc57274666)

[1. Partage en données train - test 40](#_Toc57274667)

[2. Filtres complémentaires des données 41](#_Toc57274668)

[C. IDENTIFICATION DE THEMES 42](#_Toc57274669)

[D. LABELISATION DES DOCUMENTS 43](#_Toc57274670)

[E. MODELISATION PAR Random Forests 44](#_Toc57274671)

[VII. Sources 50](#_Toc57274672)

# Introduction

## Contexte et champs d’application

Le Text Mining (Fouille de Textes) est un champ de l’apprentissage automatique appartenant lui-même au domaine de l’intelligence artificielle.

Il s’agit d’un ensemble de méthodes, techniques et outils pour exploiter des textes écrits.

Il est utilisé principalement pour

* Extraire des informations sur les textes étudiés, comme leurs mots-clés caractéristiques. Par exemple, comprendre « de quoi ils parlent » dans le cadre d’enquêtes à question(s) ouverte(s)
* Grouper des documents selon
  + leur similarité
  + leur thématique
* Retrouver des documents appartenant à une thématique donnée (*i.e.* un groupe identifié)
* Réaliser des synthèses automatiques du contenu de textes
* Analyser le sentiment plutôt positif ou négatif d’un texte (avis d’utilisateurs/consommateurs par exemple)
* …

En tant que champ du machine learning, les approches ou algorithmes d’apprentissage mis en œuvre peuvent être non supervisés ou supervisés.

## principes généraux

### Degré de structuration des données

Les données textuelles ont la particularité d’être en général « non structurées », par opposition aux tables des bases de données classiques où une ligne désigne un individu et une colonne désigne une variable mesurée/relevée par individu.

Exemples de données textuelles

|  |  |
| --- | --- |
| * non structurées |  |
| * semi-structurées |  |
| * structurées |  |

La préparation des données est donc un préalable crucial à toute analyse textuelle afin d’obtenir des données structurées, nécessaires à l’application de méthodes statistiques.

La fouille de texte comporte ainsi, le plus souvent, deux étapes majeures :

1. l’analyse, au sens de la reconnaissance des mots, des phrases, de leurs sens et relations qui permettront de caractériser un texte
2. l’interprétation de cette analyse qui permettra d’affecter le texte à un groupe (spam ou non, article traitant de spectrométrie dans une base bibliographique, …), de le sélectionner comme pertinent dans le contexte d’une recherche documentaire ou de sélectionner les phrases représentatives dans le cadre d’un résumé automatique.

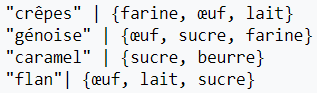
Illustration : algorithme simplifié d’indexation de textes dans le cadre d’une requête de documents

4 étapes :

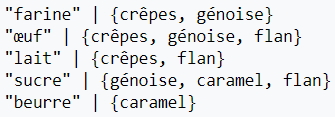
1. indexation des documents selon les mots qui les constituent
2. création de l’index inversé des mots contenus par rapport aux documents les contenant
3. Test de la similarité d’une requête de recherche documentaire avec l’index inversé
4. Renvoie des documents similaires à la requête (avec un éventuel rang de pertinence)

Exemple de mise en œuvre (source : <https://fr.wikipedia.org/wiki/Index_invers%C3%A9>) où les documents sont des recettes ramenées à leur liste d’ingrédients:

1. Index



1. Index inversé



1. Requête « farine + sucre » Þ intersection des index des mots correspondants, soit de {crêpes, génoise} et {génoise, caramel, flan}
2. Renvoi de la recette de la génoise,

"génoise" | {œuf, sucre, farine}

car la seule contenant les 2 ingrédients

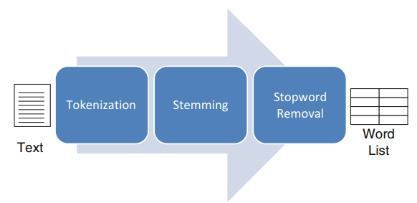
### Notion de corpus, de document et d’unité de sens

On appelle **Corpus** la collection de documents textuels que l’on souhaite analyser. Le **Document** est donc l’unité de texte dont le contenu forme « un tout », comme un article scientifique, un roman ou une simple page, selon le contexte. Enfin, l’individu statistique « élémentaire » est le mot, le plus souvent, ou l’ensemble de mots (bigram=2 mots successifs, phrase, …) formant l’unité de senssémantique qui sera utilisé dans l’analyse. On appelle cette unité jeton ou **token** en anglais.

Pour la constitution du corpus, il peut émerger en fouille de texte des problématiques éthiques ou de choix de confidentialité de l’usager (secret médical, biais dans l’autorisation ou non d’exploiter les données de navigation internet…) qui peuvent parfois perturber la représentativité des documents analysés.

# Préparation des données

Il s’agit de passer de données non structurées aux données prêtes à l’analyse statistique. La préparation initiale peut être schématisée par les 3 étapes suivantes détaillées dans ce chapitre.



* **Tokenization :** découpage d'un texte en un ensemble de jetons ou mots
  + Segmentation : diviser le texte suivant le caractère espace
  + Suppression des caractères spéciaux
  + Transformation majuscule minuscule
* **Stemming** ou **lemmatisation** : transformer les jetons identifiés précédemment en leur forme racine
* **Stopword removal :** supprimer les mots bruités ou mots vides

## Formats de données textuelles

Le format « tidy », comme en analyse de données classique, est une structure de tableau où

* Chaque ligne est une observation / un individu au sens statistique
* Chaque colonne est une variable

Dans le cadre du text mining, chaque ligne correspondra à un « jeton » (token) ; *i.e.* une unité élémentaire de texte faisant sens pour l’analyse souhaitée.

Les autres structures classiques de données textuelles sont

* Les vecteurs de chaînes de caractères
* Le corpus : ensemble de documents textes dont on souhaite faire l’analyse. Stocké comme ensemble de chaînes de caractères, annotées de détails et de métadonnées additionnelles. Chaque chaîne représente en général un document.
* La matrice « document-term » : matrice creuse décrivant une collection de textes (i.e. un corpus) où une ligne correspond à un document du corpus, une colonne à un mot. Le contenu de la matrice est typiquement un indicateur lié à la fréquence du mot dans le document.

## Préparation dans R

### Packages et workflow

Sous R, le package **tidytext** permet d’appliquer les principes du tidyverse (tidyr, dplyr, …) au text mining, afin de faciliter la préparation et la manipulation des données textuelles pour la mise en œuvre des analyses.

Sous la forme « tidy », les manipulations en R deviennent compatibles avec les outils « tidy » standards de RStudio (via dplyr, tidyr, …). Le package **tidytext** est conçu pour mettre les données textuelles sous ce format.

Les packages populaires pour l’application des algorithmes de machine learning sur données textuelles sont **tm** et **quanteda**. Ils utilisent un autre format, les matrices « document-term ». Le package **topicmodels** s’appuie également sur ce format pour la modélisation de thématiques.

Le package **hunspell** contient des fonctions utiles également, pour l’utilisation de dictionnaires de différentes langues notamment.

Le package tidytext permet le passage entre les formes tidy et les matrices document-term, les modèles obtenus peuvent être à nouveau transformés pour les interprétations et visualisations via **ggplot2** sous forme tidy.

Dans un autre ordre d’idée, le package **gutenbergr** sera utile pour accéder au contenu des 60 000 livres numériques en accès libre du [Projet Gutenberg](https://www.gutenberg.org/).

# Librairies  
library(tidytext)  
library(dplyr)  
# package de manipulation des chaînes de caractères du tidyverse  
library(stringr)   
# package graphique devenu la référence sous R  
library(ggplot2)   
# package populaire d'algorithmes de text mining  
library(tm)   
library(hunspell) # contient des fonctions de parsing, tokenization, lemmatisation...  
library(janeaustenr) # 6 romans de Jane Austen  
library(wordcloud) # pour les nuages de mots

library(forcats) # pour le graphe des tf-idf par valeur décroissante

library(gutenbergr) # accès à du contenu textuel

library(topicmodels) # modélisation de thématiques

### Tokenization

Cette étape correspond au découpage du texte en jetons. Un jeton sera le plus souvent un mot mais selon le cas, il peut aussi s’agir de n-grams (suite de n mots), de phrases ou de paragraphes.

text <- c("Premier institut de recherche agronomique en Europe et deuxième dans le monde en nombre de publications.", "Les travaux s'appliquent en sciences agricoles et en sciences de la plante et de l'animal.","L'INRA déclare mener des recherches finalisées pour une alimentation saine et de qualité, pour une agriculture durable, et pour un environnement préservé et valorisé.")  
  
text

## [1] "Premier institut de recherche agronomique en Europe et deuxième dans le monde en nombre de publications."  
## [2] "Les travaux s'appliquent en sciences agricoles et en sciences de la plante et de l'animal."  
## [3] "L'INRA déclare mener des recherches finalisées pour une alimentation saine et de qualité, pour une agriculture durable, et pour un environnement préservé et valorisé."

Ce vecteur de chaînes de caractères doit être converti en data.frame (sous son format modernisé de classe *tibble*) pour être mis sous forme de jeu de données "*tidy*", où chaque ligne sera un jeton d'un document.

# Passage en tibble  
text\_df <- tibble(line = 1:3, text = text)  
text\_df

## # A tibble: 3 x 2  
## line text   
## <int> <chr>   
## 1 1 Premier institut de recherche agronomique en Europe et deuxième dans le~  
## 2 2 Les travaux s'appliquent en sciences agricoles et en sciences de la pla~  
## 3 3 L'INRA déclare mener des recherches finalisées pour une alimentation sa~

# Tokenization  
text\_tidy <- text\_df %>%  
 unnest\_tokens(mot, text)  
text\_tidy

## # A tibble: 55 x 2  
## line mot   
## <int> <chr>   
## 1 1 premier   
## 2 1 institut   
## 3 1 de   
## 4 1 recherche   
## 5 1 agronomique  
## 6 1 en   
## 7 1 europe   
## 8 1 et   
## 9 1 deuxième   
## 10 1 dans   
## # ... with 45 more rows

Ici, nous avons un seul document. La fonction *unnest\_tokens()* coupe le texte sous forme de jetons, sur la base des espaces les séparant, et transforme le résultat en structure de données *tidy*. Les deux arguments principaux de la fonction sont

* le nom de la colonne de sortie contenant les jetons, *mot* ici et
* le nom de la colonne contenant le texte à découper, *text* dans notre tibble.

La ligne dont le mot est originaire est conservée. La ponctuation est éliminée. Par défaut, toutes les lettres sont mises en minuscules (l'argument *to\_lower=FALSE* permet de désactiver l'option). Il est possible d'utiliser d'autres modes de découpage (par lot de n-mots, par phrase, paragraphe... via l'argument *token*).

### Suppression des mots non informatifs ou « stop words »

Les « mots-outils » comme les articles le/la/les, de/du/des, les juxtapositions, les pronoms… (cf. <https://www.ranks.nl/stopwords/french>) sont des mots très fréquents dans la langue française mais n’apportent pas d’information sémantique (donc pas nonn plus sur la thématique du document).

Chaque langue a une liste donnée de « stopwords ». Ces mots seront supprimés de la liste des tokens utilisés pour les analyses subséquentes.

Le package tidytext contient un tibble nommé *stop\_words*, contenant 1149 lignes issues de 3 lexiques. Actuellement, le package ne contient en interne que les stop words de la langue anglaise.

Pour notre exemple en langue française, un tibble équivalent est créé à partir de la liste de stop words de la langue française présent dans le package **tm**.

# Liste des mots à supprimer  
stop\_words\_french <- stopwords('french')  
head(stop\_words\_french)

## [1] "au" "aux" "avec" "ce" "ces" "dans"

stop\_words\_french <- tibble(mot=stop\_words\_french, lexicon="pack\_tm")  
  
# Elimination de ces mots de nos données  
text\_tidy <- text\_tidy %>%  
 anti\_join(stop\_words\_french)

26 mots sur les 55 que nous avions initialement sont ainsi éliminés. Nous conservons pour le moment les articles avec apostrophe, comme dans "l'inra", qui ne sont pas reconnus par défaut comme des mots à part entière (puisque sans espace séparatif).

### Lemmatisation

Cette étape consiste à ramener les jetons à leur radical ou mot-racine, plusieurs formes de mots ayant la même signification en termes d’analyse textuelle.

Par exemple, un nom mis au pluriel, au féminin, sous forme d’adjectif ou les formes conjuguées d’un verbe correspondront à des tokens différents mais auront à peu de chose près le même apport sémantique.

Par défaut, RStudio n’a que des dictionnaires anglophones installés. Pour en ajouter :

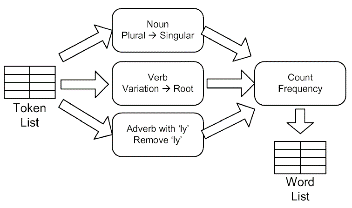
|  |  |
| --- | --- |
|  | Installation de dictionnaires de langues additionnelles (anglais seul par défaut)   * Choix de **« Install More Languages** » dans le menu déroulant |

# Suppression des "l'" et "s'"  
text\_tidy <- text\_tidy %>%  
 mutate(mot = gsub("l'","",mot),   
 mot = gsub("s'","",mot))  
# Ajout de mots manuellement au dictionnaire  
text\_tidy <- text\_tidy %>%  
 mutate(mot\_lem=unlist(sapply(hunspell\_stem(text\_tidy$mot, dict = dictionary(lang = "fr\_FR", add\_words = c("inra", "europe"))), function(xx){xx[[1]]})))

# Affichage des 6 premières lignes du tableau  
knitr::kable(head(text\_tidy))

|  |  |  |
| --- | --- | --- |
| line | mot | mot\_lem |
| 1 | premier | première |
| 1 | institut | institut |
| 1 | recherche | recherche |
| 1 | agronomique | agronomique |
| 1 | europe | europe |
| 1 | deuxième | deuxième |

L'argument *lang* permet de préciser le dictionnaire utiliser pour la lemmatisation (Français ici). L'argument *add\_words* de la fonction *dictionary()* permet d'ajouter des mots manuellement, s'ils ne sont pas reconnus par le dictionnaire utilisé. La lemmatisation via *hunspell\_stem()* peut renvoyer plusieurs propositions, seule la première est considérée ici.



# Analyse descriptive

## Fréquence des mots

Il s’agit de compter le nombre d’occurrences de chaque jeton identifié dans le corpus.

text\_tidy\_count <- text\_tidy %>%  
 count(mot\_lem, sort=T)

knitr::kable(head(text\_tidy\_count))

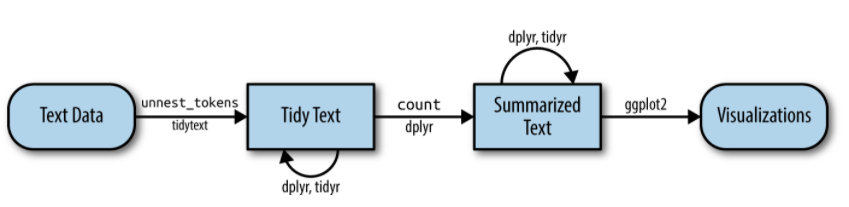
|  |  |
| --- | --- |
| mot\_lem | n |
| recherche | 2 |
| science | 2 |
| agricole | 1 |
| agriculture | 1 |
| agronomique | 1 |
| alimentation | 1 |

Ce premier exemple contient peu de matière. Ces étapes sont répétées sur un **corpus** plus large de 6 romans de Jane Austen, compilés dans le package **janeaustenr**.

Les étapes sont:

1. Lecture des données, le format adopté par le package est "une ligne du tibble = une ligne du livre publié". Il est possible également d'identifier les différents chapitres via l'usage de *regex()*, fonction de **R** permettant la recherche dans les *expressions régulières*.
2. Mise au format tidy via *unnest\_tokens()*
3. Suppression des "stop words"
4. Calcul de la fréquence des mots
5. Visualisations

Le workflow sous R peut être représenté comme suit :



Source : Text Mining with R (J.Silge & D Robinson)

# Etape 1: données d'origine  
original\_books <- austen\_books() %>%  
 group\_by(book) %>%  
 mutate(linenumber = row\_number(),  
 chapter = cumsum(str\_detect(text,   
 regex("^chapter [\\divxlc]",  
 ignore\_case = TRUE)))) %>%  
 ungroup()  
  
original\_books

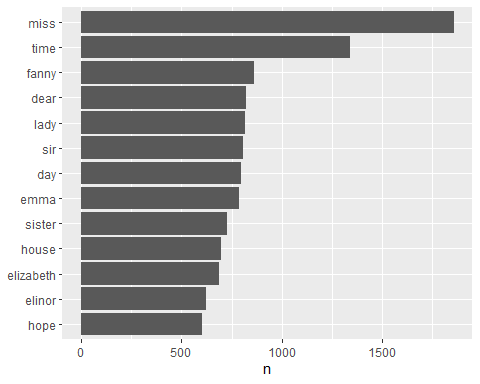
|  |  |  |  |
| --- | --- | --- | --- |
| text | book | linenumber | chapter |
| SENSE AND SENSIBILITY | Sense & Sensibility | 1 | 0 |
|  | Sense & Sensibility | 2 | 0 |
| by Jane Austen | Sense & Sensibility | 3 | 0 |
|  | Sense & Sensibility | 4 | 0 |
| (1811) | Sense & Sensibility | 5 | 0 |
|  | Sense & Sensibility | 6 | 0 |

# Etape 2: format tidy  
tidy\_books <- original\_books %>%  
 unnest\_tokens(word, text)  
tidy\_books

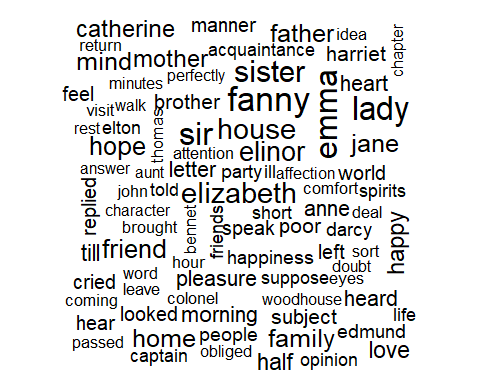
|  |  |  |  |
| --- | --- | --- | --- |
| book | linenumber | chapter | word |
| Sense & Sensibility | 1 | 0 | sense |
| Sense & Sensibility | 1 | 0 | and |
| Sense & Sensibility | 1 | 0 | sensibility |
| Sense & Sensibility | 3 | 0 | by |
| Sense & Sensibility | 3 | 0 | jane |
| Sense & Sensibility | 3 | 0 | austen |

# Etape 3: suppression des stopwords  
data(stop\_words)  
  
tidy\_books <- tidy\_books %>%  
 anti\_join(stop\_words)  
  
# Etape 4: calcul des fréquences  
janeausten\_freq <- tidy\_books %>%  
 count(word, sort = TRUE) %>%

mutate(word = reorder(word, n))   
  
# Etape 5: Visualisation des mots à >600 occurences  
  
# --- Graphe en barres  
janeausten\_freq %>%  
 filter(n > 600) %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(n, word)) +  
 geom\_col() +  
 labs(y = NULL)



# --- Nuage de mots  
janeausten\_freq %>%  
 with(wordcloud(word, n, max.words = 100))



## Statistique tf-idf

### Définition

Une question centrale en fouille de textes est de quantifier le sujet d’un document donné. La fréquence des mots est une partie de la réponse, une fois la question des « stop words » prise en compte.

Une alternative est de pondérer ces fréquences des mots par l’inverse de leur fréquence à travers la collection de documents (i.e. le corpus).

Ainsi, un ensemble d’articles traitant de différents thèmes, dont le sport et la science par exemple, pourrait témoigner de la fréquence élevée des mots « match » dans certains articles et « chercheur » dans d’autres, selon leur thème respectif. Contrairement aux stop words abondants partout, ces mots seraient abondants dans certains documents (fréquence élevée pour ces mots), mais ne concerner qu’un nombre restreint de documents au sein du corpus étudié (faible fréquence de documents).

Ces mots fréquents dans un petit nombre de documents ont une probabilité élevée d’être ceux révélant deux thèmes distincts.

La statistique de synthèse de ces deux fréquences est appelée **tf-idf** (term frequency-inverse document frequency). Sa fomule est

Où

* ni,j : le nombre d’apparitions du mot i dans le document j
* nk,j: le nombre d’apparitions du mot k dans le document j
* C : le nombre de mots différents dans le document j
* NDocuments : le nombre total de documents dans le corpus
* Nj : le nombre de documents contenant le mot i au moins une fois

Il n’y a pas de base théorique établie derrière cette statistique en théorie de l’information, mais elle a prouvé son utilité à l’usage.

La valeur du tf-idf sera élevée si le mot est fréquent dans certains documents, mais que les documents le contenant sont rares. Le terme sera alors d’autant plus caractéristique, donc discriminant, du contenu de ces documents par rapport aux autres que le tf\_idf est élevé.

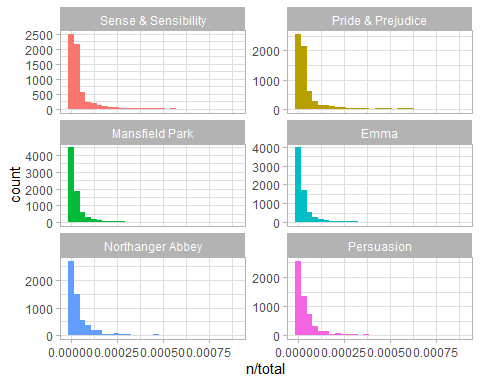
### Illustration

Sur le corpus des romans de Jane Austen : cette fois, les mots vont être comptés par *document*, c'est à dire par livre. Le nombre total de mots par roman sera également relevé. Nous obtiendrons ainsi le premier indice **TF** ou **Term Frequency** en divisant ces deux quantités, *i.e.* la proportion d'apparition de chaque combinaison *mot-roman*.

# book\_words <- tidy\_books %>%  
# count(book, word, sort = TRUE)  
#   
book\_words <- austen\_books() %>%  
 unnest\_tokens(word, text) %>%  
 count(book, word, sort = TRUE)  
  
total\_words <- book\_words %>%   
 group\_by(book) %>%   
 summarize(total = sum(n))  
  
book\_words <- left\_join(book\_words, total\_words) %>%  
 mutate(tf=n/total)  
  
# --- TF des romans  
knitr::kable(head(book\_words))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| book | word | n | total | tf |
| Mansfield Park | the | 6206 | 160460 | 0.0386763 |
| Mansfield Park | to | 5475 | 160460 | 0.0341207 |
| Mansfield Park | and | 5438 | 160460 | 0.0338901 |
| Emma | to | 5239 | 160996 | 0.0325412 |
| Emma | the | 5201 | 160996 | 0.0323052 |
| Emma | and | 4896 | 160996 | 0.0304107 |

# --- Graphique par roman  
ggplot(book\_words, aes(n/total, fill = book)) +  
 geom\_histogram(show.legend = FALSE) +  
 xlim(NA, 0.0009) +  
 facet\_wrap(~book, ncol = 2, scales = "free\_y") +   
 theme\_light()



Nous constatons graphiquement une distribution avec une très longue queue de distribution à droite : certains mots sont extrêmement fréquents, certains seront très rares. Ceci est typique de tout corpus de langage naturel.

**Remarque**: dans les différentes langues, il s'avère que la relation entre le TF et le rang du mot en termes de TF est linéaire sur l'échelle log-log (loi de **Zipf**).

book\_tf\_idf <- book\_words %>%  
 bind\_tf\_idf(word, book, n)  
  
knitr::kable(head(book\_tf\_idf))

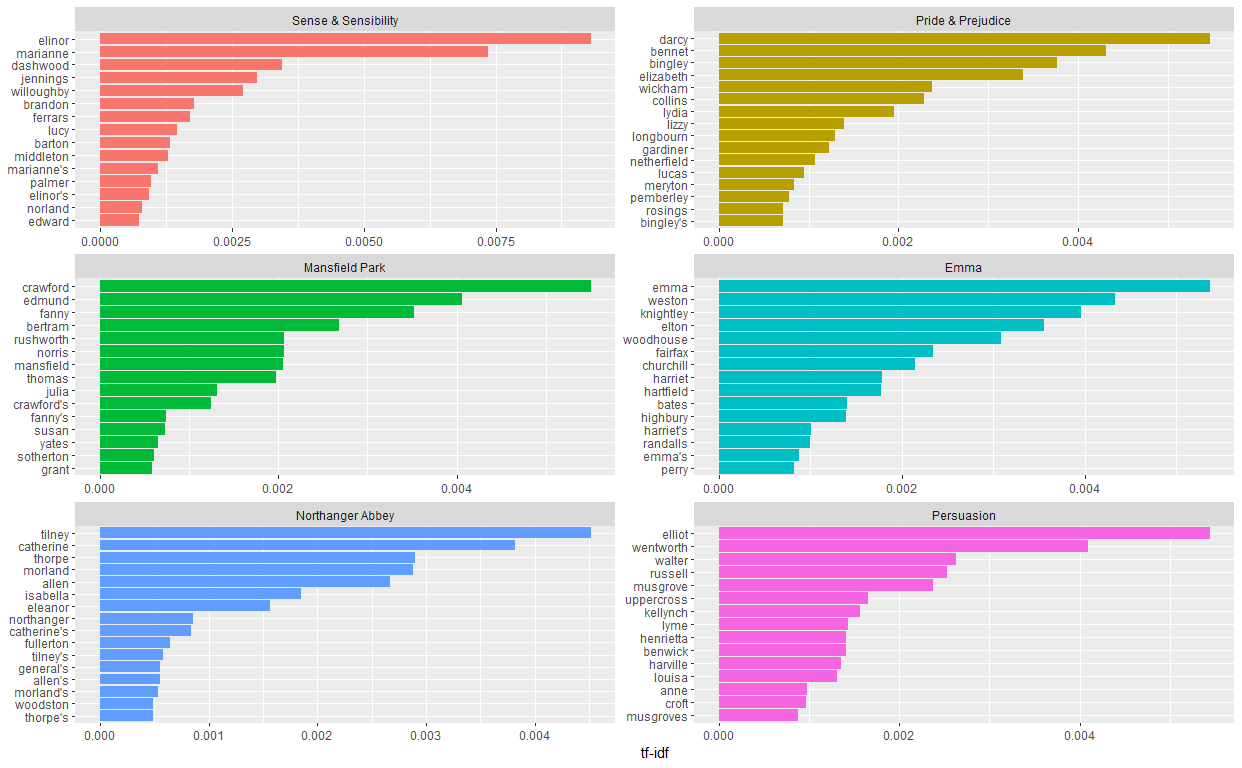
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| book | word | n | total | tf | idf | tf\_idf |
| Mansfield Park | the | 6206 | 160460 | 0.0386763 | 0 | 0 |
| Mansfield Park | to | 5475 | 160460 | 0.0341207 | 0 | 0 |
| Mansfield Park | and | 5438 | 160460 | 0.0338901 | 0 | 0 |
| Emma | to | 5239 | 160996 | 0.0325412 | 0 | 0 |
| Emma | the | 5201 | 160996 | 0.0323052 | 0 | 0 |
| Emma | and | 4896 | 160996 | 0.0304107 | 0 | 0 |

Les mots non informatifs (stop words) sont automatiquement filtrés : présents dans tous les documents, leur **idf** est nulle et donc leur **tf-idf** également.

# Tri par valeur de TF-IDF  
book\_tf\_idf %>%  
 dplyr::select(-total) %>%  
 arrange(desc(tf\_idf)) %>%  
 head(n=10) %>%  
 knitr::kable()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| book | word | n | tf | idf | tf\_idf |
| Sense & Sensibility | elinor | 623 | 0.0051935 | 1.791759 | 0.0093056 |
| Sense & Sensibility | marianne | 492 | 0.0041015 | 1.791759 | 0.0073488 |
| Mansfield Park | crawford | 493 | 0.0030724 | 1.791759 | 0.0055050 |
| Pride & Prejudice | darcy | 373 | 0.0030523 | 1.791759 | 0.0054689 |
| Persuasion | elliot | 254 | 0.0030362 | 1.791759 | 0.0054401 |
| Emma | emma | 786 | 0.0048821 | 1.098612 | 0.0053635 |
| Northanger Abbey | tilney | 196 | 0.0025199 | 1.791759 | 0.0045151 |
| Emma | weston | 389 | 0.0024162 | 1.791759 | 0.0043293 |
| Pride & Prejudice | bennet | 294 | 0.0024058 | 1.791759 | 0.0043106 |
| Persuasion | wentworth | 191 | 0.0022831 | 1.791759 | 0.0040908 |

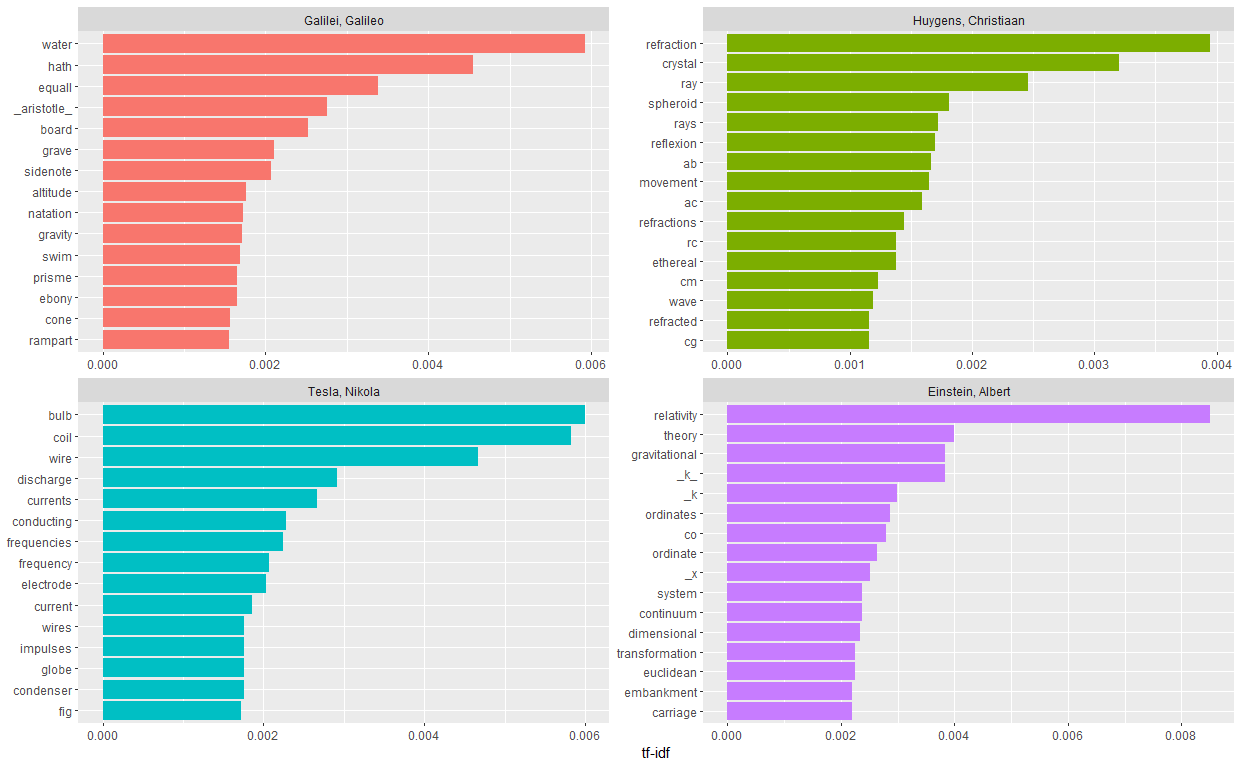
# Graphique  
book\_tf\_idf %>%  
 group\_by(book) %>%  
 slice\_max(tf\_idf, n = 15) %>%  
 ungroup() %>%  
 ggplot(aes(tf\_idf, fct\_reorder(word, tf\_idf), fill = book)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~book, ncol = 2, scales = "free") +  
 labs(x = "tf-idf", y = NULL)



Dans le cadre de romans comme ici, les noms propres correspondant aux personnages principaux particuliers à chaque roman sont effectivement attendus comme particulièrement discriminants.

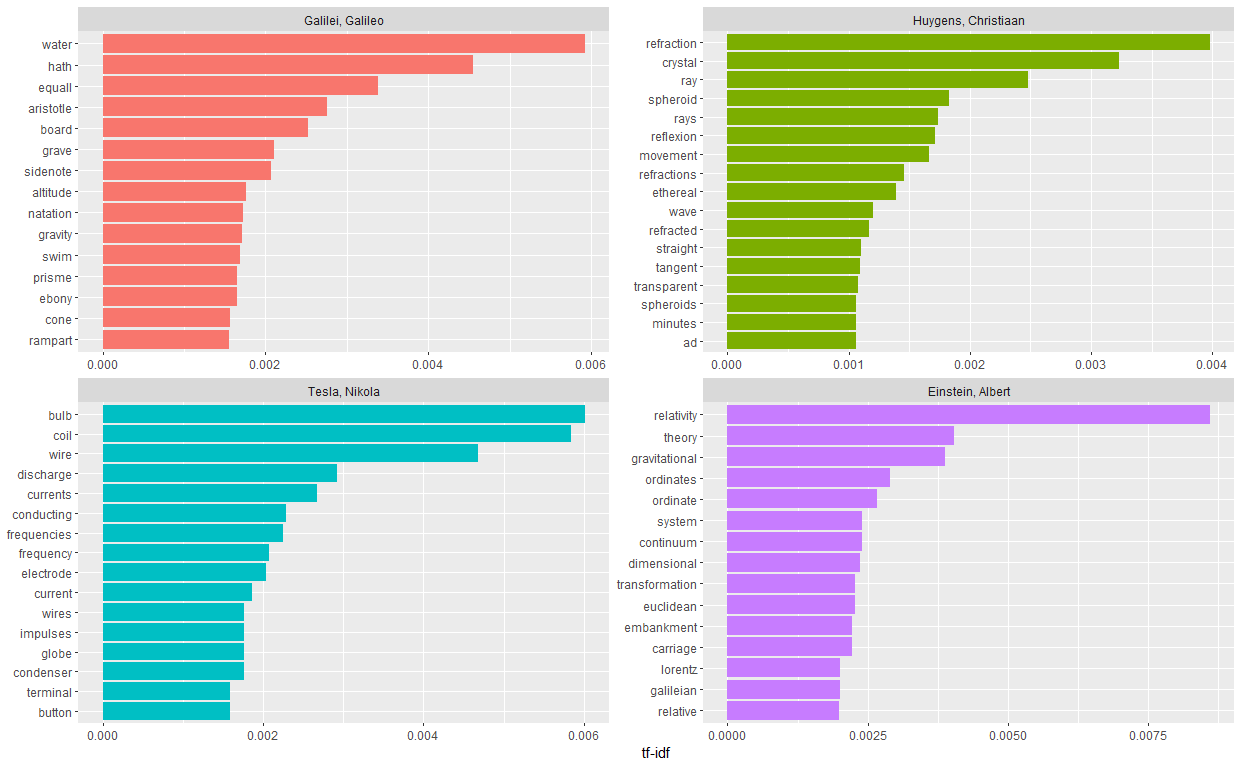
Afin d'identifier ces mots discriminants dans un autre contexte, réalisons un second exemple sur des articles scientifiques en sciences physiques issus du Projet Gutenberg.

# Lecture des données  
physics <- gutenberg\_download(c(37729, 14725, 13476, 30155),   
 meta\_fields = "author")  
  
# Préparation  
physics\_words <- physics %>%  
 unnest\_tokens(word, text) %>%  
 count(author, word, sort = TRUE)  
  
# Graphique des tf-idf  
physics\_words %>%  
 # --- Ajout du tf-idf et filtre/ordonnancement des données  
 bind\_tf\_idf(word, author, n) %>%  
 mutate(author = factor(author, levels = c("Galilei, Galileo",  
 "Huygens, Christiaan",   
 "Tesla, Nikola",  
 "Einstein, Albert"))) %>%   
 group\_by(author) %>%   
 slice\_max(tf\_idf, n = 15) %>%   
 ungroup() %>%  
 mutate(word = reorder(word, tf\_idf)) %>%  
 # --- Graphe  
 ggplot(aes(tf\_idf, word, fill = author)) +  
 geom\_col(show.legend = FALSE) +  
 labs(x = "tf-idf", y = NULL) +  
 facet\_wrap(~author, ncol = 2, scales = "free")



Des termes surprenants apparaissent, comme *ab*, *rc*... pour Huygens, correspondant à des noms d'angles, rayons, cercles... *k* chez Einstein, la séparation de *co* et *ordinate* car séparés par un caractère de ponctuation... un nettoyage complémentaire peut donc être nécessaire pour clarifier l'interprétation.

# tibble de stopwords complémentaires   
mystopwords <- tibble(word = c("eq", "co", "rc", "ac", "ak", "bn",   
 "fig", "file", "cg", "cb", "cm",  
 "ab", "\_k", "\_k\_", "\_x"))  
# Suppression de ces mots  
physics\_words <- physics\_words %>%  
 anti\_join(mystopwords, by = "word")  
  
physics\_words %>%  
 # --- Ajout du tf-idf et filtre/ordonnancement des données  
 bind\_tf\_idf(word, author, n) %>%  
 mutate(word = str\_remove\_all(word, "\_")) %>%  
 group\_by(author) %>%   
 slice\_max(tf\_idf, n = 15) %>%  
 ungroup() %>%  
 mutate(word = reorder\_within(word, tf\_idf, author)) %>%  
 mutate(author = factor(author, levels = c("Galilei, Galileo",  
 "Huygens, Christiaan",  
 "Tesla, Nikola",  
 "Einstein, Albert"))) %>%  
 # --- Graphe  
 ggplot(aes(word, tf\_idf, fill = author)) +  
 geom\_col(show.legend = FALSE) +  
 labs(x = NULL, y = "tf-idf") +  
 facet\_wrap(~author, ncol = 2, scales = "free") +  
 coord\_flip() +  
 scale\_x\_reordered()



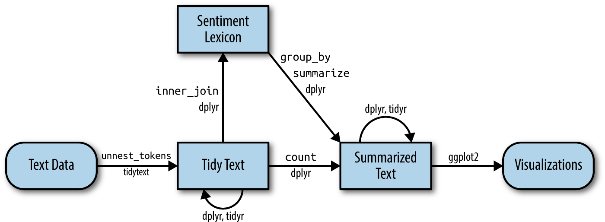
L'usage des indicateurs **TF** et **TF-IDF** permet respectivement d'explorer l'usage de la langue dans un corpus donné et de trouver les mots caractéristiques de chaque document dans ce corpus. La mise au format *TIDY* des données en amont permet de conserver une logique d'exploration et d'analyse proche de celle des données "classiques".

# Analyse de sentiment (Sentiment analysis)

Ce champ de la fouille de texte vise à comprendre le contenu émotionnel et les opinions exprimés dans les documents analysés. L’approche permet d’identifier les séquences de texte « positives », « négatives » ou exprimant des émotions plus nuancées, comme la surprise.

Une approche possible, couramment utilisée mais pas unique, est de considérer le texte comme une combinaison de ses mots et son contenu émotionnel global comme la somme de ceux de ses jetons individuels (mots ou n-grams).

Le workflow sous R peut être schématisé comme suit :



Source : Text Mining with R (J.Silge & D Robinson)

## Les dictionnaires de sentiment

Comme évoqué précédemment, plusieurs méthodes et plusieurs dictionnaires existent pour évaluer l’opinion ou l’émotion portées par un texte. Ces dictionnaires affectent un score ou un qualificatif à chaque mot « informatif » de la langue.

3 dictionnaires principaux en anglais, obtenus par la fonction *get\_sentiments()* de **tidytext**. Ils doivent être téléchargés lors de la première utilisation, ce qui nécessite d'installer le package **textdata**.

get\_sentiments("afinn")

## # A tibble: 2,477 x 2  
## word value  
## <chr> <dbl>  
## 1 abandon -2  
## 2 abandoned -2  
## 3 abandons -2  
## 4 abducted -2  
## 5 abduction -2  
## 6 abductions -2  
## 7 abhor -3  
## 8 abhorred -3  
## 9 abhorrent -3  
## 10 abhors -3  
## # ... with 2,467 more rows

get\_sentiments("bing")

## # A tibble: 6,786 x 2  
## word sentiment  
## <chr> <chr>   
## 1 2-faces negative   
## 2 abnormal negative   
## 3 abolish negative   
## 4 abominable negative   
## 5 abominably negative   
## 6 abominate negative   
## 7 abomination negative   
## 8 abort negative   
## 9 aborted negative   
## 10 aborts negative   
## # ... with 6,776 more rows

get\_sentiments("nrc")

## # A tibble: 13,901 x 2  
## word sentiment  
## <chr> <chr>   
## 1 abacus trust   
## 2 abandon fear   
## 3 abandon negative   
## 4 abandon sadness   
## 5 abandoned anger   
## 6 abandoned fear   
## 7 abandoned negative   
## 8 abandoned sadness   
## 9 abandonment anger   
## 10 abandonment fear   
## # ... with 13,891 more rows

Les points de vigilance à avoir en tête :

* Ces dictionnaires sont le fruit d'un travail manuel, validé par la communauté. Ce travail est essentiellement réalisé sur des avis ou message internet d'aujourd'hui, avec leurs spécificités.
* L'approche sera ici basée sur des mots (unigrams), donc le sarcasme ou la négation, tels que dans *no good* ou *not so wonderful* ne seront pas pris en compte.
* Enfin, la taille des sections de texte sur lesquelles se feront les aggrégations de sentiments de mots individuels, phrase, paragraphe, chapitre... peut impacter le résultat. Plus la section est longue, plus le risque est grand de finir sur une moyenne proche de 0 ; alors que les trop petites peuvent manquer de mots pour estimer correctement l'émotion.

## Le sentiment à travers un corpus de texte

L'illustration sera celle des romans de Jane Austen, bien que ces textes soient des fictions d'environ 200 ans, pas les plus en rapport avec les tweets d'aujourd'hui...

# Données  
tidy\_books <- austen\_books() %>%  
 group\_by(book) %>%  
 mutate(  
 linenumber = row\_number(),  
 chapter = cumsum(str\_detect(text,   
 regex("^chapter [\\divxlc]",   
 ignore\_case = TRUE)))) %>%  
 ungroup() %>%  
 unnest\_tokens(word, text)

L'utilisation du nom de colonne "*word*" pour contenir les jetons est pratique pour les jointures avec les dictionnaires, ayant eux-mêmes ce nom pour le vecteur de mots.

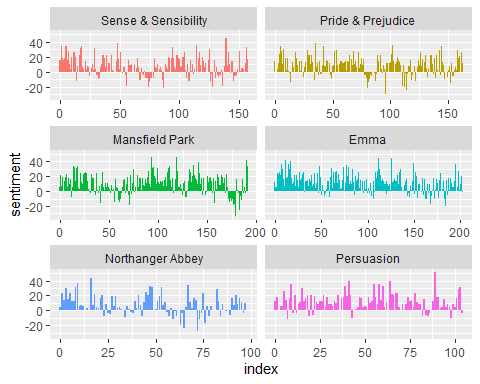
# Mots "joyeux" les plus fréquents dans EMMA  
nrc\_joy <- get\_sentiments("nrc") %>%   
 filter(sentiment == "joy")  
  
tidy\_books %>%  
 filter(book == "Emma") %>%  
 inner\_join(nrc\_joy) %>%  
 count(word, sort = TRUE)

## # A tibble: 303 x 2  
## word n  
## <chr> <int>  
## 1 good 359  
## 2 young 192  
## 3 friend 166  
## 4 hope 143  
## 5 happy 125  
## 6 love 117  
## 7 deal 92  
## 8 found 92  
## 9 present 89  
## 10 kind 82  
## # ... with 293 more rows

De nombreux mots sont effectivement positifs, en lien avec l'amitié, l'amour, l'espoir. D'autres comme "*found*" ou "*present*" peuvent avoir plusieurs sens et certains ne sont pas spécialement joyeux.

Etendons ce type d'étude aux 6 romans, en utilisant un autre dictionnaire, *Bing* basé sur un score. L'unité d'aggrégation de score sera une suite de 80 lignes. Un index permettra de suivre la position de chaque section dans les livres.

# Calcul du sentiment positif ou négatif par section  
jane\_austen\_sentiment <- tidy\_books %>%  
 inner\_join(get\_sentiments("bing")) %>%  
 count(book, index = linenumber %/% 80, sentiment) %>%  
 spread(sentiment, n, fill = 0) %>%  
 mutate(sentiment = positive - negative)  
  
# Visualisation par roman  
ggplot(jane\_austen\_sentiment, aes(index, sentiment, fill = book)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~book, ncol = 2, scales = "free\_x")



# --- Représentation des deux pôles + et –

jane\_austen\_sentiment\_long <- jane\_austen\_sentiment %>%

pivot\_longer(cols = negative:positive,

names\_to="polarite",  
 values\_to="score")  
ggplot(jane\_austen\_sentiment\_long, aes(index, score, fill = book)) +

geom\_col(show.legend = FALSE) +

facet\_wrap(~book+polarite, ncol = 2, scales = "free\_x")

jane\_austen\_sentiment\_long <- jane\_austen\_sentiment %>%

pivot\_longer(cols = negative:sentiment,

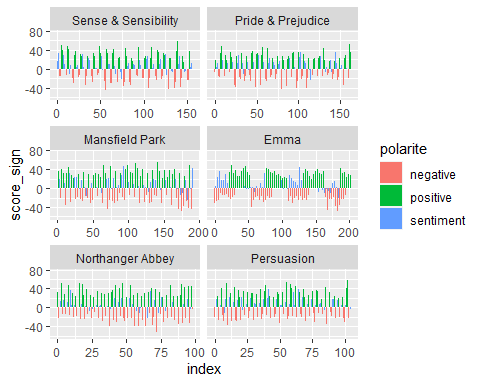
names\_to="polarite",  
 values\_to="score") %>%

mutate(score\_sign=ifelse(polarite=="negative", -score, score))

ggplot(jane\_austen\_sentiment\_long, aes(index, score\_sign, fill = polarite)) +

geom\_col(show.legend = T, position = "dodge") +

facet\_wrap(~book, ncol = 2, scales = "free\_x")



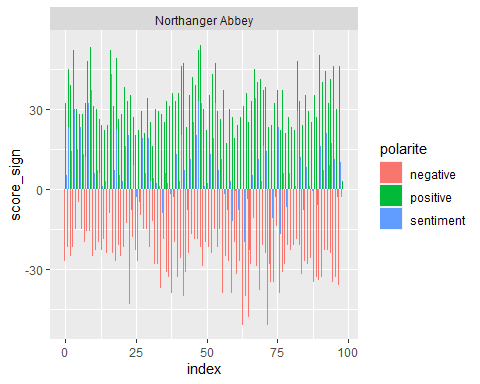
# Représentation de la triple information (+, -, sentiment synthétique = différence des 2)

# Barres juxtaposées

ggplot(jane\_austen\_sentiment\_long %>% dplyr::filter(book=="Northanger Abbey"), aes(index, score\_sign, fill = polarite)) +

geom\_col(show.legend = T, position = "dodge") +

facet\_wrap(~book, ncol = 2, scales = "free\_x")



# Barres empilées

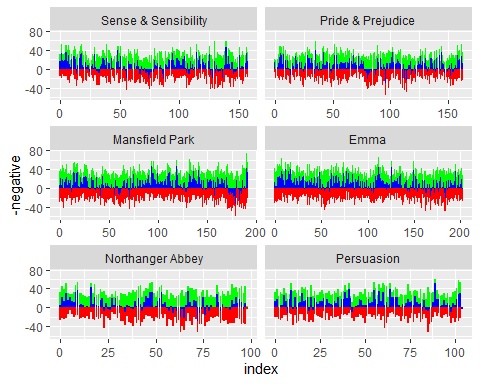
ggplot(jane\_austen\_sentiment, aes(index, -negative)) +

geom\_col(show.legend = T, fill="red") +

geom\_col(aes(index, positive), fill = "green") +

geom\_col(aes(index, sentiment), fill="blue") +

facet\_wrap(~book, ncol = 2, scales = "free\_x")



## L’analyse des contribution au sentiment

Ayant les mots des romans et le sentiment associé à disposition, nous pouvons rechercher les mots contribuant le plus au sens de l'émotion identifiée comme positive ou négative.

### Mots positifs et négatifs les plus fréquents

bing\_word\_counts <- tidy\_books %>%  
 inner\_join(get\_sentiments("bing")) %>%  
 count(word, sentiment, sort = TRUE) %>%  
 ungroup()

bing\_word\_counts

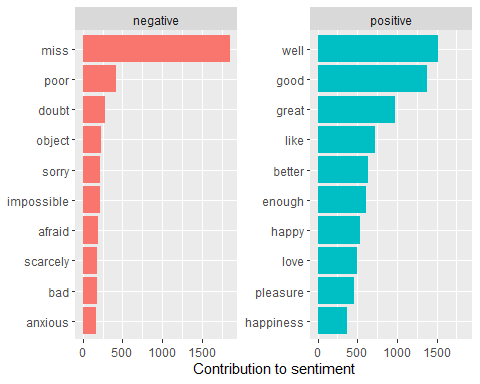
## # A tibble: 2,585 x 3  
## word sentiment n  
## <chr> <chr> <int>  
## 1 miss negative 1855  
## 2 well positive 1523  
## 3 good positive 1380  
## 4 great positive 981  
## 5 like positive 725  
## 6 better positive 639

### Représentations graphiques

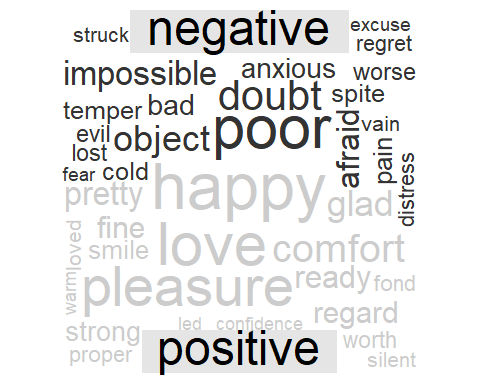
Les barplots classiques permettent de classer les mots les plus fréquents selon le sentiment positif ou négatif et se faire identifier ainsi les responsables principaux de l'émotion générale.

La fonction *comparison.cloud()* permet d'obtenir un wordcloud organisant les mots selon l'émotion portée, positive ou négative. Ceci demande un retour à un objet de classe *matrix* réalisé par la fonction *acast()* du package **reshape2**.

# Barplots  
bing\_word\_counts %>%  
 group\_by(sentiment) %>%  
 top\_n(10) %>%  
 ungroup() %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(n, word, fill = sentiment)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~sentiment, scales = "free\_y") +  
 labs(x = "Contribution to sentiment",  
 y = NULL)



# Ajustement manuel des mots au sens mal identifié  
custom\_stop\_words <- bind\_rows(tibble(word = c("miss"),   
 lexicon = c("custom")),   
 stop\_words)  
  
# Wordcloud  
xx <- tidy\_books %>%  
 inner\_join(get\_sentiments("bing")) %>%  
 anti\_join(custom\_stop\_words) %>%  
 count(word, sentiment, sort = TRUE) %>%  
 acast(word ~ sentiment, value.var = "n", fill = 0) %>%  
 comparison.cloud(colors = c("gray20", "gray80"),  
 max.words = 100)



## Bilan

La connaissance des livres permet de valider ou tempérer les résultats obtenus. Ils s'avèrent plutôt pertinents.

**Pour aller plus loin**, nous pourrions analyser l'émotion positive ou négative à des échelles supérieures au mot. Ceci permet des analyses plus fines et évite les pièges grossiers qui identifieraient "Ce n'est vraiment pas super." comme une phrase globalement positive grâce au *super* présent.

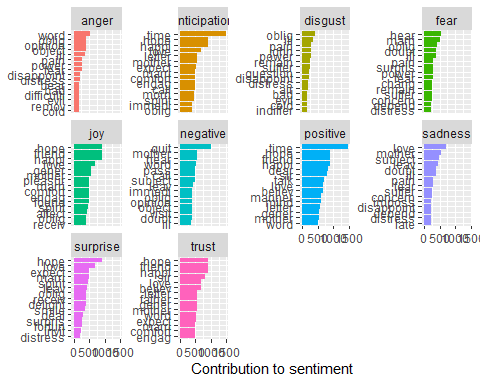
## Complément sur la lemmatisation et le sentiment

tidy\_books\_lemm <- tidy\_books %>%  
 mutate(word\_ini=word, word=SnowballC::wordStem(word))  
  
lex\_nrc\_lemm <- get\_sentiments("nrc") %>%  
 mutate(word\_ini=word, word=SnowballC::wordStem(word)) %>%  
 dplyr::select(-word\_ini) %>%  
 distinct()  
  
tmp <- get\_sentiments("nrc")[grep("feel", get\_sentiments("nrc")$word), ]  
   
custom\_stop\_words <- bind\_rows(tibble(word = c("miss", "of", "feel"),   
 lexicon = c("custom")),   
 stop\_words)  
nrc\_word\_counts <- tidy\_books\_lemm %>%  
 inner\_join(lex\_nrc\_lemm, by="word") %>%  
 count(word, sentiment, sort = TRUE) %>%  
 ungroup()

nrc\_word\_counts

## # A tibble: 6,811 x 3  
## word sentiment n  
## <chr> <chr> <int>  
## 1 of negative 21178  
## 2 will anger 2269  
## 3 will negative 2269  
## 4 will sadness 2269  
## 5 miss fear 1905  
## 6 miss negative 1905  
## 7 miss sadness 1905  
## 8 know positive 1534  
## 9 time anticipation 1474  
## 10 time positive 1474  
## # ... with 6,801 more rows

# Barplots  
nrc\_word\_counts %>%  
 anti\_join(custom\_stop\_words) %>%  
 group\_by(sentiment) %>%  
 top\_n(15) %>%  
 ungroup() %>%  
 mutate(word = reorder(word, n)) %>%  
 ggplot(aes(n, word, fill = sentiment)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~sentiment, scales = "free\_y") +  
 labs(x = "Contribution to sentiment",  
 y = NULL)



# Wordcloud

tidy\_books\_lemm %>%

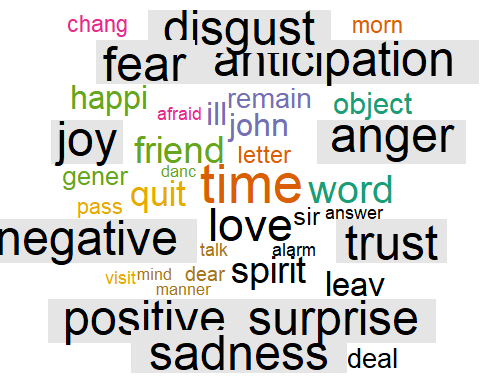
inner\_join(lex\_nrc\_lemm, by="word") %>%

anti\_join(custom\_stop\_words) %>%

count(word, sentiment, sort = TRUE) %>%

acast(word ~ sentiment, value.var = "n", fill = 0) %>%

comparison.cloud(max.words = 100)



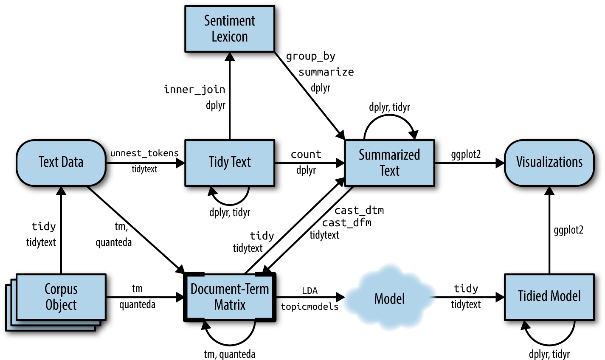
# Modélisation de la thématique

## Contexte et workflow

Disposant d’un corpus, il est classique en fouille de texte de vouloir le diviser en groupes afin de comprendre de quoi traite chacun. Il s’agit d’une approche « non supervisée » similaire à la classification (i.e. **clustering** en anglais), cherchant à identifier les groupes « naturels » sans avoir d’à priori sur ce qu’ils contiennent.

La LDA (Latent Dirichlet Allocation) est une méthode courante d’ajustement de modèle de thématique. L’approche autorise les chevauchements entre documents en termes de contenu, plutôt que de les séparer en groupes cloisonnés. La suite de ce chapitre illustre l’apprentissage des différences entre documents, basé sur leur contenu.

Comme l’illustre le workflow suivant, les méthodes précédentes de préparation de données sont reprises, travaillant sur une structure « tidy », mais un passage à la structure « matrice de termes » sera nécessaire pour construire le modèle. Celui-ci sera à son tour mis sous forme « tidy » pour son interprétation et visualisation.



Source : Text Mining with R (J.Silge & D Robinson)

## Approche LDA

L’approche est fondée sur deux principes

* Chaque document est un mélange de sujets (thèmes), chacun pouvant être représenté dans des proportions différentes (exemple : article à 80% sportif et 20% scientifique).
* Chaque thème est un mélange de mots (exemple : deux thèmes, la politique et les loisirs, sur un site d’actualités. « Président », « assemblée » et « gouvernement » seraient les mots les plus communs du premier thème ; tandis que « série », « acteur » et « cinéma » seraient les plus fréquents du second. Un mot comme « budget » pourrait être également représenté dans les deux thèmes.)

La LDA estime les deux choses à la fois :

* Quel mélange de mots est associé à chaque thème,
* Quel mélange de thèmes décrit au mieux chaque document

La mise en œuvre est effectuée sur des dépêches d'actualités issues de l'**Associated Press**.

## Ajustement de la LDA

Le modèle est créé par la fonction *LDA()* de **topicmodels**. L'argument *k = 2* signifie qu'un modèle à 2 thèmes seulement est créé, mais ceci s'étend à un nombre plus important en pratique. Une graine de nombres aléatoires, via l’argument *seed*, est fournie pour permettre d'avoir des résultats répétables.

# Ajustement du modèle  
data("AssociatedPress")  
ap\_lda <- LDA(AssociatedPress, k = 2, control = list(seed = 1234))  
ap\_lda

## A LDA\_VEM topic model with 2 topics.

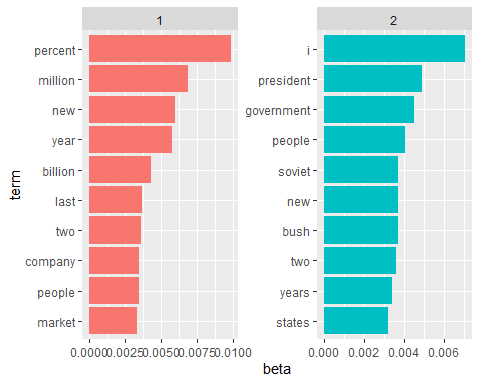
## Obtention des probabilités "par-mot-par-thème"

La mise au format *tidy* du modèle par la fonction *tidy()* de **tidytext** permet d'obtenir facilement les probabilités par thème que chacun des mots en soit issu, notées , via l'argument *matrix="beta"*. Celles-ci sont très basses pour de nombreux mots. La fonction *top\_n()* de dplyr permet de trouver les 10 mots les plus communs à chaque thème et d'en faire le barplot via **ggplot2**.

# Obtention des probabilités "par mot et par thème", notées "beta"  
ap\_topics <- tidy(ap\_lda, matrix = "beta")  
knitr::kable(head(ap\_topics))

|  |  |  |
| --- | --- | --- |
| topic | term | beta |
| 1 | aaron | 0.0000000 |
| 2 | aaron | 0.0000390 |
| 1 | abandon | 0.0000265 |
| 2 | abandon | 0.0000399 |
| 1 | abandoned | 0.0001391 |
| 2 | abandoned | 0.0000588 |

# liste des 10 mots les plus communs par thème et barplots  
ap\_top\_terms <- ap\_topics %>%  
 group\_by(topic) %>%  
 top\_n(10, beta) %>%  
 ungroup() %>%  
 arrange(topic, -beta)  
  
ap\_top\_terms %>%  
 mutate(term = reorder\_within(term, beta, topic)) %>%  
 ggplot(aes(beta, term, fill = factor(topic))) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~ topic, scales = "free") +  
 scale\_y\_reordered()



## Interprétation des thèmes

Le rapide parcours de ces deux listes de mots laissent à penser que le premier thème est en rapport avec le monde des affaires, le second en rapport avec la politique. Ce second thème doit d'ailleurs s'appuyer sur des données plutôt anciennes, mais probablement fortement représentées dans l'historique de la base, les relations USA-USSR étant un gros sujet politique durant des décennies.

**Remarque**: les mots "new" et "people" sont communs aux deux thèmes. Ils n'ont pas été plus ou moins arbitrairement affectés à l'un ou l'autre thème, comme l'aurait fait une classification "dure" (hard clustering de la CAH ou des K-means). Ceci constitue un **important avantage** de la méthode : des chevauchements sont possibles, en accord avec l'usage du langage naturel.

## Compléments à l'interprétation

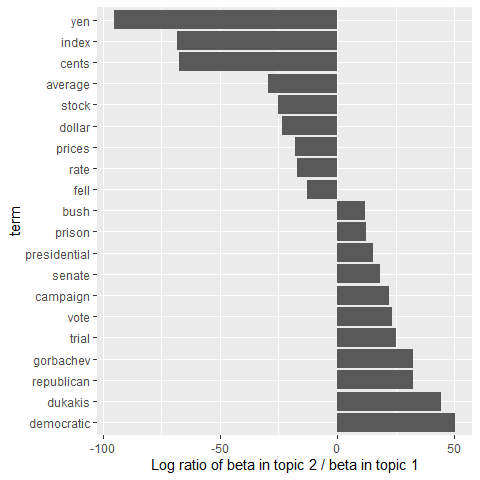
### Différences principales entre les thèmes

Il est possible de rechercher les *plus grandes différences* en termes de entre les deux thèmes. Elles sont estimées sur la base du rapport des log des probabilités d'appartenir à l'un ou l'autre thème , pratique car rendant les différences symétriques entre thèmes.

# Tableau  
beta\_spread <- ap\_topics %>%  
 mutate(topic = paste0("topic", topic)) %>%  
 spread(topic, beta) %>%  
 filter(topic1 > .001 | topic2 > .001) %>%  
 mutate(log\_ratio = log2(topic2 / topic1))  
  
knitr::kable(head(beta\_spread, n=10))

|  |  |  |  |
| --- | --- | --- | --- |
| term | topic1 | topic2 | log\_ratio |
| administration | 0.0004310 | 0.0013822 | 1.6814189 |
| ago | 0.0010652 | 0.0008421 | -0.3390353 |
| agreement | 0.0006715 | 0.0010390 | 0.6297728 |
| aid | 0.0000476 | 0.0010460 | 4.4580091 |
| air | 0.0021369 | 0.0002967 | -2.8486628 |
| american | 0.0020305 | 0.0016839 | -0.2700405 |
| analysts | 0.0010876 | 0.0000006 | -10.8778386 |
| area | 0.0013714 | 0.0002310 | -2.5695069 |
| army | 0.0002622 | 0.0010481 | 1.9989152 |
| asked | 0.0001886 | 0.0015592 | 3.0475641 |

# Graphe des 20 plus grosses différences  
plot\_log.ratio <- beta\_spread %>%  
 top\_n(20, abs(log\_ratio)) %>%  
 arrange(desc(log\_ratio))   
  
plot\_log.ratio %>%  
 ggplot(aes(log\_ratio, term)) +  
 geom\_col(show.legend = F) +  
 ylim(plot\_log.ratio$term) +  
 labs(x="Log ratio of beta in topic 2 / beta in topic 1")



Le thème 1 est plutôt caractérisé par des noms de monnaies et des termes financiers, comme "index", "prices" ou "rates". Le second thème présente parmi ses termes les plus communs les noms de partis américains ou de politiciens, russes ou américains. Ceci confirme que l'algorithme a identifié les thèmes financier et politique dans cette collection d'articles d'actualités.

### Proportions de chaque thème dans les documents

Le modèle LDA représente également chaque document comme un mélange de thèmes. Nous obtenons des estimations des probabilités "par thème par document", notées , via l'argument *matrix="gamma"*.

ap\_documents <- tidy(ap\_lda, matrix = "gamma")  
knitr::kable(head(ap\_documents, n=10))

|  |  |  |
| --- | --- | --- |
| document | topic | gamma |
| 1 | 1 | 0.2480617 |
| 2 | 1 | 0.3615485 |
| 3 | 1 | 0.5265844 |
| 4 | 1 | 0.3566530 |
| 5 | 1 | 0.1812767 |
| 6 | 1 | 0.0005883 |
| 7 | 1 | 0.7734216 |
| 8 | 1 | 0.0044517 |
| 9 | 1 | 0.9669915 |
| 10 | 1 | 0.1468905 |

Chaque probabilité est une estimation de la proportion de mots du document issue de chacun des thèmes. D'après le tableau ci-dessus, le document 2 aurait un peu plus d'un tiers de ses termes issus du thème 1 "financier". **Remarque**: Par construction, où i est l'un des topic identifié, quelque soit le nombre k fournit à la LDA.

Le document 6 a une proportion très faible de mots issus du thème 1, on peut penser qu'il est essentiellement lié au second thème "politique", quand le document 7 doit porter principalement sur la finance et les affaires.

validation <- tidy(AssociatedPress) %>%  
 filter(document == 6 | document==7) %>%  
 group\_by(document) %>%  
 arrange(desc(count)) %>%  
 ungroup()  
  
knitr::kable(validation[validation$document == 6,][1:10,])

|  |  |  |
| --- | --- | --- |
| document | term | count |
| 6 | noriega | 16 |
| 6 | panama | 12 |
| 6 | jackson | 6 |
| 6 | powell | 6 |
| 6 | administration | 5 |
| 6 | economic | 5 |
| 6 | general | 5 |
| 6 | i | 5 |
| 6 | panamanian | 5 |
| 6 | american | 4 |

knitr::kable(validation[validation$document == 7,][1:10,])

|  |  |  |
| --- | --- | --- |
| document | term | count |
| 7 | skins | 9 |
| 7 | japan | 6 |
| 7 | trade | 5 |
| 7 | american | 3 |
| 7 | cites | 3 |
| 7 | documents | 3 |
| 7 | export | 3 |
| 7 | group | 3 |
| 7 | illegal | 3 |
| 7 | international | 3 |

knitr::kable(validation[validation$document == 1271,][1:10,])

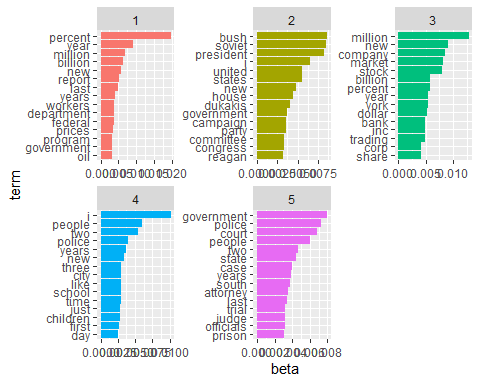
|  |  |  |
| --- | --- | --- |
| document | term | count |
| 1271 | index | 4 |
| 1271 | rsqb | 4 |
| 1271 | stock | 4 |
| 1271 | friday | 3 |
| 1271 | last | 3 |
| 1271 | market | 3 |
| 1271 | american | 2 |
| 1271 | big | 2 |
| 1271 | blue | 2 |
| 1271 | boeing | 2 |

knitr::kable(validation[validation$document == 2000,][1:10,])

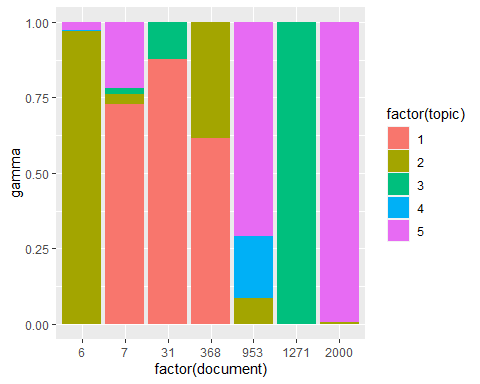
|  |  |  |
| --- | --- | --- |
| document | term | count |
| 2000 | owen | 19 |
| 2000 | cocaine | 8 |
| 2000 | spain | 8 |
| 2000 | barahona | 7 |
| 2000 | suitcase | 7 |
| 2000 | drug | 6 |
| 2000 | case | 4 |
| 2000 | charges | 4 |
| 2000 | court | 4 |
| 2000 | drugs | 4 |

## Exploration de l'impact de k sur la distribution des gammas par document

# Ajustement LDA  
ap\_lda <- LDA(AssociatedPress, k = 5, control = list(seed = 1234))  
ap\_documents <- tidy(ap\_lda, matrix = "gamma")  
# str(ap\_lda)  
  
# Calcul des betas  
ap\_topics <- tidy(ap\_lda, matrix = "beta")  
  
# liste des 10 mots les plus communs par thème et barplots  
ap\_top\_terms <- ap\_topics %>%  
 group\_by(topic) %>%  
 top\_n(15, beta) %>%  
 ungroup() %>%  
 arrange(topic, -beta)  
  
ap\_top\_terms %>%  
 mutate(term = reorder\_within(term, beta, topic)) %>%  
 ggplot(aes(beta, term, fill = factor(topic))) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~ topic, scales = "free") +  
 scale\_y\_reordered()



# Visualisation des gammas pour quelques documents  
ap\_documents %>%  
 dplyr::filter(document %in% c(6,7,368,31, 953, 2000, 1271)) %>%  
 ggplot(aes(x=factor(document), y=gamma, group=document, fill=factor(topic)))+  
 geom\_col(position = "fill")



# Approche supervisée

# Approche issue de <https://cfss.uchicago.edu/notes/supervised-text-classification/#weighting>

# settings  
set.seed(1234)  
theme\_set(theme\_minimal())  
  
ap\_tidy <- tidytext::tidy(AssociatedPress)

## Préparation des données

### Stemming des termes de AssociatedPress

ap\_tidy\_stem <- ap\_tidy %>%  
 mutate(word=SnowballC::wordStem(term))  
  
# Valeur de l'IDF correspondant à des mots présents dans >=75% des documents  
docs\_N <- nDocs(AssociatedPress) #ap\_tidy\_stem\_dtm  
seuil <- log(docs\_N/round(0.2\*docs\_N))  
# Filtre des mots sur la base du tf-idf, mais en revenant à une dtm pondérée en Term Freq pour permettre la LDA  
ap\_tidy\_stem\_idf <- ap\_tidy\_stem %>%   
 bind\_tf\_idf(word, document, count) %>%  
 dplyr::filter(idf>=seuil)  
  
# summary(ap\_tidy\_stem\_idf$idf)  
# hist(ap\_tidy\_stem\_idf$idf)

### Reconstruction de la matrice Document-Term

ap\_tidy\_stem\_dtm <- ap\_tidy\_stem\_idf %>%  
 count(document, word) %>%  
 cast\_dtm(document = document, term = word, value = n) # Argument weighting pour pondérer les mots par tf-idf au lieu de tf (par défaut): weighting = tm::weightTfIdf)   
  
# Comparaison des 2 matrices  
AssociatedPress

## <<DocumentTermMatrix (documents: 2246, terms: 10473)>>  
## Non-/sparse entries: 302031/23220327  
## Sparsity : 99%  
## Maximal term length: 18  
## Weighting : term frequency (tf)

ap\_tidy\_stem\_dtm

## <<DocumentTermMatrix (documents: 2246, terms: 6825)>>  
## Non-/sparse entries: 244509/15084441  
## Sparsity : 98%  
## Maximal term length: 18  
## Weighting : term frequency (tf)

## Identification et allocation des documents à différents thèmes

### Partage en données train - test

test\_prop <- 0.3  
docs\_ID <- Docs(ap\_tidy\_stem\_dtm)  
# ID des documents qui seront conservés en jeu "test"  
set.seed(742) # 2chantillonnage du jeu de test sera "répétable"  
docs\_test <- sample(seq(1, docs\_N), round(0.3\*docs\_N))  
  
# Test de sélection train/test sur la matrice DT directement   
# (OK, mais si pondération tf-idf, elle considère aussi les données test par cette approche)  
ap\_dtm\_train <- ap\_tidy\_stem\_dtm[-docs\_test,]  
ap\_dtm\_train

## <<DocumentTermMatrix (documents: 1572, terms: 6825)>>  
## Non-/sparse entries: 167960/10560940  
## Sparsity : 98%  
## Maximal term length: 18  
## Weighting : term frequency (tf)

ap\_dtm\_test <- ap\_tidy\_stem\_dtm[docs\_test,]  
ap\_dtm\_test

## <<DocumentTermMatrix (documents: 674, terms: 6825)>>  
## Non-/sparse entries: 76549/4523501  
## Sparsity : 98%  
## Maximal term length: 18  
## Weighting : term frequency (tf)

# Pour utiliser une pondération TF-IDF issue du jeu "train" uniquement:  
# ap\_stem.filterSparse\_train <- tidytext::tidy(ap\_tidy\_stem\_dtm) %>%  
# dplyr::select(-count) %>%  
# dplyr::filter(!(document %in% docs\_test)) %>%  
# count(document, term) %>%  
# cast\_dtm(document = document, term = term, value = n, weighting = tm::weightTfIdf)  
  
# Pour le jeu test:   
# - Extraire les pondérations issue de la matrice train - revenir au format tidy pour matrice test - affecter les les poids tfidf issus de matrice train aux mots de matrice test et retour au format dtm (document term matrix)  
# ap\_stem.filterSparse\_test <- tidytext::tidy(ap\_tidy\_stem\_dtm) %>%  
# ... étape d'affectation des poids

### Filtres complémentaires des données

Mots présents dans trop peu de documents

ap\_dtm\_train\_filtreSparse <- removeSparseTerms(ap\_dtm\_train, sparse = .95)  
ind\_cols\_0 <- colSums(as.matrix(ap\_dtm\_train\_filtreSparse))!=0  
table(ind\_cols\_0)

## ind\_cols\_0  
## TRUE   
## 556

ind\_rows\_0 <- rowSums(as.matrix(ap\_dtm\_train\_filtreSparse))!=0  
table(ind\_rows\_0)

## ind\_rows\_0  
## FALSE TRUE   
## 1 1571

which(ind\_rows\_0==F)

## 709   
## 485

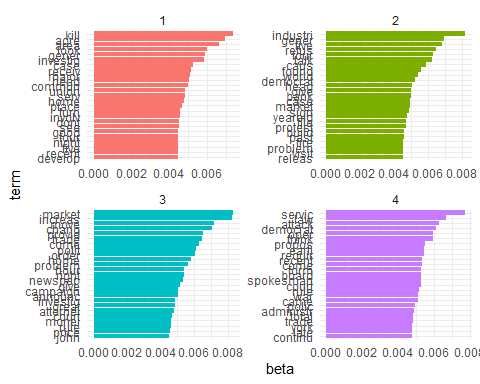
sum(as.matrix(ap\_dtm\_train\_filtreSparse)[485,])

## [1] 0

Documents n'ayant plus de mots après le removeSparse (à effectuer en amont du découpage train/test?)  
  
ap\_dtm\_train\_filtreSparse <- ap\_dtm\_train\_filtreSparse[-485,]

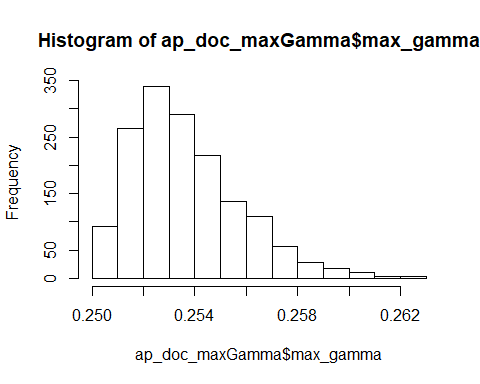
## IDENTIFICATION DE THEMES

# Ajustement LDA  
# --- sur données d'entrainement  
ap\_lda\_train <- LDA(ap\_dtm\_train\_filtreSparse, k = 4, control = list(seed = 1234))  
# --- sur données complètes  
# ap\_lda\_train <- LDA(ap\_tidy\_stem\_dtm, k = 5, control = list(seed = 1234))  
  
ap\_documents <- tidy(ap\_lda\_train, matrix = "gamma")  
# str(ap\_lda\_train)  
  
# Calcul des betas  
ap\_topics <- tidy(ap\_lda\_train, matrix = "beta")  
  
# liste des 10 mots les plus communs par thème et barplots  
ap\_top\_terms <- ap\_topics %>%  
 group\_by(topic) %>%  
 top\_n(25, beta) %>%  
 ungroup() %>%  
 arrange(topic, -beta)  
  
ap\_top\_terms %>%  
 mutate(term = reorder\_within(term, beta, topic)) %>%  
 ggplot(aes(beta, term, fill = factor(topic))) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~ topic, scales = "free") +  
 scale\_y\_reordered()



## LABELISATION DES DOCUMENTS

ap\_doc\_maxGamma <- ap\_documents %>%  
 group\_by(document) %>%  
 summarise(max\_gamma=max(gamma))  
  
hist(ap\_doc\_maxGamma$max\_gamma)



ap\_doc\_wide <- ap\_documents %>%  
 pivot\_wider(names\_from = topic, values\_from=gamma)  
  
# Recherche de la colonne correspondant au gamma max => numéro du topic  
id\_topic <- function(xx, data=ap\_doc\_wide){  
 vec\_gamma <-ap\_doc\_wide[data$document==xx,-1]  
 return(topic=which(vec\_gamma==max(vec\_gamma)))  
}  
  
topic <- factor(sapply(ap\_doc\_wide$document, id\_topic, data=ap\_doc\_wide))   
topic.test <- factor(sapply(ap\_doc\_wide$document, id\_topic, data=ap\_doc\_wide))

## MODELISATION PAR Random Forests

# K-folds CV  
mtry <- c(10, 20, 50)  
ntree <- c(300, 400, 500)  
  
seed.ini=482  
n.folds <- 10  
folds.cc <- createFolds(y=topic,k=n.folds,list=TRUE)  
sapply(folds.cc,length)

## Fold01 Fold02 Fold03 Fold04 Fold05 Fold06 Fold07 Fold08 Fold09 Fold10   
## 157 157 158 158 158 157 156 156 157 157

res\_RF <- vector(mode="list", length(ntree)\*length(mtry))  
  
system.time(  
 for (ii in 1:length(ntree)){  
 for (jj in 1:length(mtry)){  
 tab\_perf <- NULL  
 for (zz in 1:n.folds) {  
 set.seed(seed.ini+zz+1)  
 rf.train <- randomForest(x=as.matrix(ap\_dtm\_train\_filtreSparse)[-folds.cc[[zz]],], y=topic[-folds.cc[[zz]]],   
 mtry=mtry[jj],   
 ntree=ntree[ii])  
 test.model = predict(rf.train, type = "resp", newdata = as.matrix(ap\_dtm\_train\_filtreSparse)[folds.cc[[zz]],])  
 res <- cbind.data.frame(doc=names(test.model),   
 pred=test.model,  
 label=topic[folds.cc[[zz]]])  
 tab\_perf <- rbind.data.frame(tab\_perf, res)  
 }  
 res\_RF[[(ii-1)\*length(mtry)+jj]] <- tab\_perf  
 }  
 })

## user system elapsed   
## 1779.14 2.20 1782.69

# Exploration des résultats  
head(res\_RF[[1]])

## doc pred label  
## 4 4 3 1  
## 27 27 4 4  
## 37 37 4 4  
## 62 62 2 1  
## 66 66 3 4  
## 67 67 3 3

# str(res\_RF)  
  
# nom de l'élément=combinaison de ntree/mtry correspondante  
names(res\_RF) <- paste(rep(paste0("ntree.", ntree), each=length(mtry)), rep(paste0("mtry.", mtry), times=length(ntree)))  
  
# Calcul des accuracy (indicateur de performance) par combinaison d'hyperparamètres mtry/ntree  
sapply(res\_RF, function(xx){  
 mat\_conf <- table(xx$pred, xx$label)  
 return(accuracy=round(sum(diag(mat\_conf))/dim(xx)[1]\*100,2))  
})

## ntree.300 mtry.10 ntree.300 mtry.20 ntree.300 mtry.50 ntree.400 mtry.10   
## 51.75 51.43 52.39 52.13   
## ntree.400 mtry.20 ntree.400 mtry.50 ntree.500 mtry.10 ntree.500 mtry.20   
## 51.81 53.28 52.45 52.32   
## ntree.500 mtry.50   
## 53.09

# Valeurs optimales des hyperparamètres identifiées  
mtry <- 10  
ntree <- 500  
  
# Entrainement du modèle sur l'ensemble du jeu d'entrainement  
system.time(model <- randomForest(x=as.matrix(ap\_dtm\_train\_filtreSparse), y=topic, mtry=mtry, ntree=ntree, na.action = na.omit))

## user system elapsed   
## 27.89 0.01 27.90

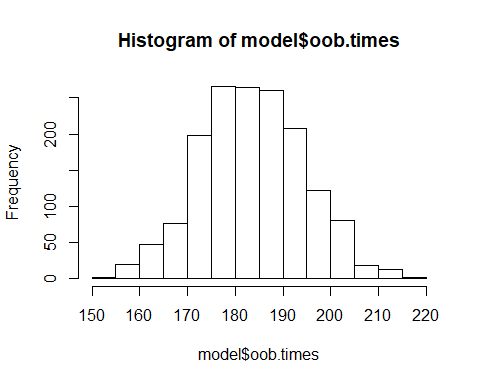
# Exploration des résultats du modèle optimal  
# --- Prédiction des données du jeu test  
pred\_test = predict(model, type = "resp", newdata = as.matrix(ap\_dtm\_test))  
  
# Attention ici, les labels du jeu de test ne sont pas directement accessibles, il faudrait pour cela recalculer les gamma de la LDA sur ces données test.  
# UNe fois disponible, les commandes suivantes donneraient une estimation honnête de l'accuracy du modèle final  
res\_jeu.test <- cbind.data.frame(doc=names(pred\_test),   
 pred=pred\_test,  
 label=topic[docs\_test])  
  
mat\_conf <- table(res\_jeu.test $pred, res\_jeu.test $label)  
  
# Qualité estimée du modèle final "de production"  
round(sum(diag(mat\_conf))/dim(xx)[1]\*100,2)

## numeric(0)

model

##   
## Call:  
## randomForest(x = as.matrix(ap\_dtm\_train\_filtreSparse), y = topic, ntree = ntree, mtry = mtry, na.action = na.omit)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 10  
##   
## OOB estimate of error rate: 49.01%  
## Confusion matrix:  
## 1 2 3 4 class.error  
## 1 272 25 54 61 0.3398058  
## 2 119 85 73 66 0.7521866  
## 3 99 25 241 49 0.4178744  
## 4 102 21 76 203 0.4950249

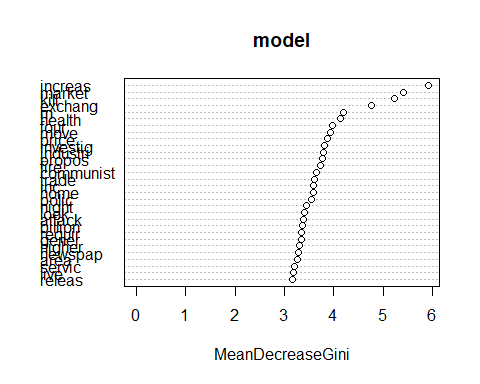
hist(model$oob.times)



model$votes[1:10,]

## 1 2 3 4  
## 4 0.24444444 0.2111111 0.2500000 0.29444444  
## 7 0.25373134 0.2388060 0.2537313 0.25373134  
## 8 0.26063830 0.3244681 0.3191489 0.09574468  
## 9 0.48633880 0.1475410 0.1857923 0.18032787  
## 10 0.32608696 0.2065217 0.2173913 0.25000000  
## 11 0.45454545 0.1497326 0.3208556 0.07486631  
## 15 0.06030151 0.2311558 0.5427136 0.16582915  
## 16 0.20725389 0.3108808 0.1813472 0.30051813  
## 17 0.33146067 0.3089888 0.2752809 0.08426966  
## 18 0.28160920 0.2758621 0.2241379 0.21839080

# Hiérarchie des prédicteurs  
impo\_words <- model$importance  
varImpPlot(model,type=2)



# Entrainement final du modèle avec les mêmes valeurs d'hyperparamètres sur l'ensemble du Corpus (les 2246 docs ici), modèle qui sera effectivement utilisé pour prédire les nouvelles données entrantes  
  
# Filtre des mots présents dans trop peu de documents:  
ap\_dtm\_filtreSparse <- removeSparseTerms(ap\_tidy\_stem\_dtm, sparse = .95)  
ind\_cols\_0 <- colSums(as.matrix(ap\_dtm\_filtreSparse))!=0  
table(ind\_cols\_0)

## ind\_cols\_0  
## TRUE   
## 575

# Indices des documents contenant toujours des mots à l'issu du filtre  
ind\_rows\_0 <- rowSums(as.matrix(ap\_dtm\_filtreSparse))!=0  
table(ind\_rows\_0)

## ind\_rows\_0  
## FALSE TRUE   
## 1 2245

which(ind\_rows\_0==F)

## 709   
## 709

sum(as.matrix(ap\_dtm\_filtreSparse)[!ind\_rows\_0,])

## [1] 0

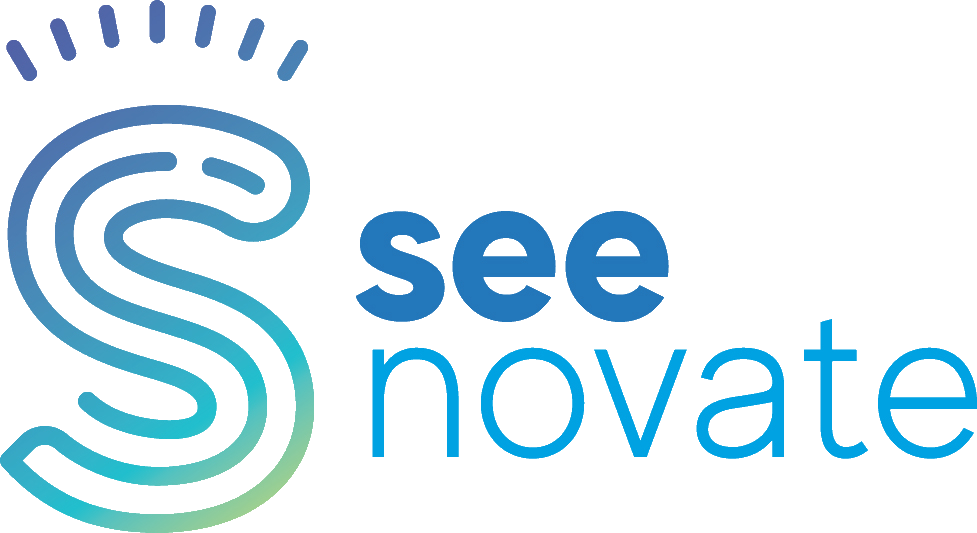
# Filtre des docs n'ayant plus de mots après le removeSparse à effectuer en amont du découpage train/test?  
  
# Données complètes, documents non vides à l'issue du filtre "removeSparse"  
ap\_dtm\_train\_filtreSparse <- ap\_dtm\_train\_filtreSparse[ind\_rows\_0,]  
  
# Après labellisation des données test sur la base de la LDA, par exemple, ajustement du modèle final de production:  
# system.time(model\_final.production <- randomForest(x=as.matrix(ap\_dtm\_filtreSparse), y=topic, mtry=mtry, ntree=ntree, na.action = na.omit))

# Sources

* Wikipédia
* <https://www.tidytextmining.com/index.html>
* <https://arxiv.org/pdf/1707.02919.pdf>
* <https://perso.uclouvain.be/vincent.blondel/publications/08-textmining.pdf>
* <http://data-bzh.fr/text-mining-r-part-1/> à <http://data-bzh.fr/text-mining-r-part-3/>
* <http://www.sthda.com/french/wiki/text-mining-et-nuage-de-mots-avec-le-logiciel-r-5-etapes-simples-a-savoir>
* <https://www.springboard.com/blog/text-mining-in-r/>
* <https://medium.com/text-mining-in-data-science-a-tutorial-of-text/text-mining-in-data-science-51299e4e594>

**Divers :**

* Parsing de BibTex <https://cran.r-project.org/web/packages/bib2df/vignettes/bib2df.html>
* Problème accès dictionnaires <https://github.com/ropensci/hunspell/issues/3>
* Page du CRAN consacrée aux outils de NLP <https://cran.r-project.org/web/views/NaturalLanguageProcessing.html>
* N-grams :
  + <https://bookdown.org/Maxine/tidy-text-mining/tokenizing-by-n-gram.html>
  + <https://cran.r-project.org/web/packages/ngram/vignettes/ngram-guide.pdf>
  + Outils dans quanteda?
* [Text2Vec package in R](http://text2vec.org/glove.html)
* [Traitement du langage naturel en Deep Learning](https://blog.paperspace.com/6-interesting-deep-learning-applications-for-nlp/)
* [Text CLustering en utilisant du deep learning (Python)](https://towardsdatascience.com/making-sense-of-text-clustering-ca649c190b20)



<https://bit.ly/2MKWwzJ>www.seenovate.com

**PARIS**

11 Rue Dulong

75017 Paris

**LYON**

8 Rue de la Bourse

69002 Lyon

**MONTPELLIER**

Immeuble MIBI 672

Rue du Mas de Verchant

34000 Montpellier

**RENNES**

BURO Club Rennes-Sud

Place du Granier BP 97143

35571 Chantepie Cedex