Traitement des questions ouvertes avec R

Paul BALAFAI et Mame Balla BOUSSO

2025-04-19

Plan de présentation

- Traitement des questions ouvertes avec R
  - 1. Importation et Nettoyage des Données
  - 2. Exploration et Prétraitement Textuel
  - 3. Analyse Thématique (LDA)
  - 4. Approche Alternative avec BERTopic
  - 5. Catégorisation
  - CONCLUSION
- Références

INTRODUCTION

Le traitement automatique du langage naturel (TALN) regroupe un ensemble de techniques permettant d'analyser, de comprendre et de transformer des textes en données exploitables.

La forme la plus courante est l'analyse **supervisée**, où chaque texte est associé à un label prédéfini. Ces labels peuvent par exemple représenter des catégories binaires comme 0 ou 1, ou encore des sentiments comme positif, négatif ou neutre.

Cependant, dans de nombreux cas, notamment dans les **questions ouvertes d'enquêtes**, il n'existe aucune annotation préalable permettant de guider l'apprentissage. Il devient alors nécessaire de structurer les données sans repère préalable, en regroupant les textes selon leur **similarité sémantique**. Dans cette étude, nous nous concentrerons spécifiquement sur cette approche **non supervisée**.

Packages : **topicmodels (pour le modèle LDA)**, **tidytext**, et **BERTopic** (**reticulate**) .

Importation des données

package

```
library(haven)
library(readxl)          # Pour lire les fichiers Excel
library(topicmodels)     # Pour la modélisation thématique
library(ggplot2)         # Pour les visualisations
library(dplyr)           # Pour la manipulation de données
library(tidytext)        # Pour le traitement de texte
library(tidyr)           # Pour la gestion des données
library(wordcloud)       # Pour les nuages de mots
library(tidyverse)       # Collection de packages pour la science
library(tm)              # Pour le text mining
library(SnowballC)       # Pour le stemming
library(stringr)         # Pour la manipulation de strings
```

Importation des données

```
## # A tibble: 6 x 4
##      id ‘Classe de l’étudiant :‘ ‘Nationalité de l’étudiant
##   <dbl> <chr>                    <chr>
## 1     1 AS2                      Congo
## 2     2 ISEP1                    Cameroun
## 3     3 AS2                      Congo
## 4     4 AS1                      Sénégal
## 5     5 AS1                      Cameroun
## 6     6 ISE1 Eco                 Sénégal
##  [1] "id"           "E01"          "E01_AUTRE"   "E02"          "E
##  [6] "E03A_AUTRE" "E03B"         "E03B_AUTRE" "E04"          "E
## [1] 1945697
```

base rgph

```
## [1] Maison basse
## [2] Case
## [3] Maison à étages R+1
## [4] Autre
## [5] Baraque
## [6] Maison à étages R+2
## [7] Maison à étages R+3
## [8] Maison à étages R+4
## [9] Appartement dans un immeuble (R+5 ou plus)
## 9 Levels: Case Baraque Maison basse ... Autre
```

Vérification des colonnes

```
## [1] "id"                                "Classe de l'étudiant :"
## [3] "Nationalité de l'étudiant" "Texte"
```

Identification des textes vides

```
## # A tibble: 6 x 4
##      id `Classe de l'étudiant :` `Nationalité de l'étudiant
##   <dbl> <chr>                    <chr>
## 1     2 ISEP1                    Cameroun
## 2     4 AS1                      Sénégal
## 3     5 AS1                      Cameroun
## 4     7 ISE2                     Cameroun
## 5     9 ISEP2                    Cameroun
## 6    11 ISE2                     Togo
## [1] 45
```

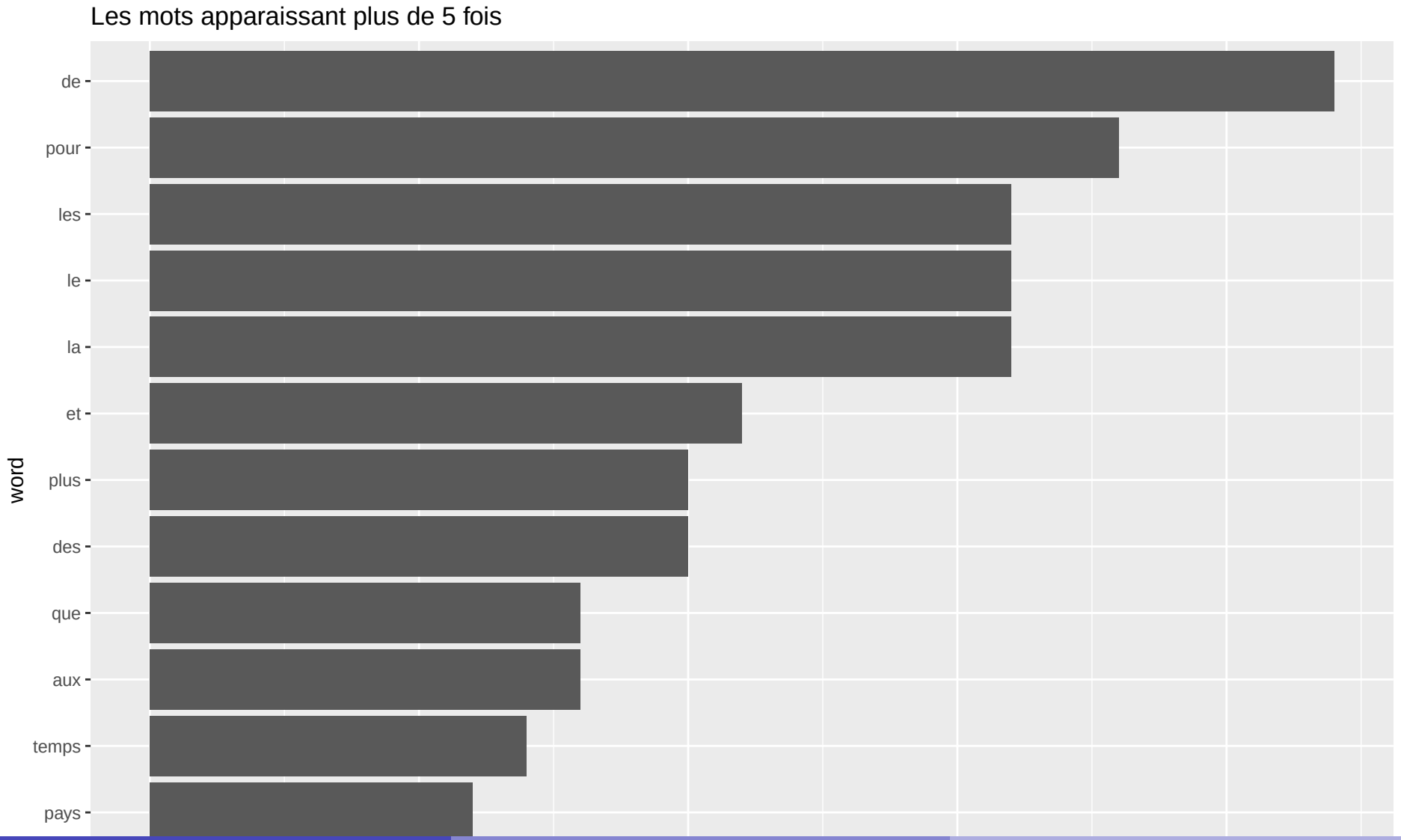
On remarque que le nombre de ligne a diminué passant de 128 à 45. Seulement 45 lignes contiennent des textes.

Nettoyage des textes

On crée une fonction pour traiter les textes afin de faciliter leur analyse

2. Exploration et Prétraitement des textes

Dans le processus de prétraitement des données, on va tokeniser la base de données pour analyser non pas les textes, mais les mots directement.

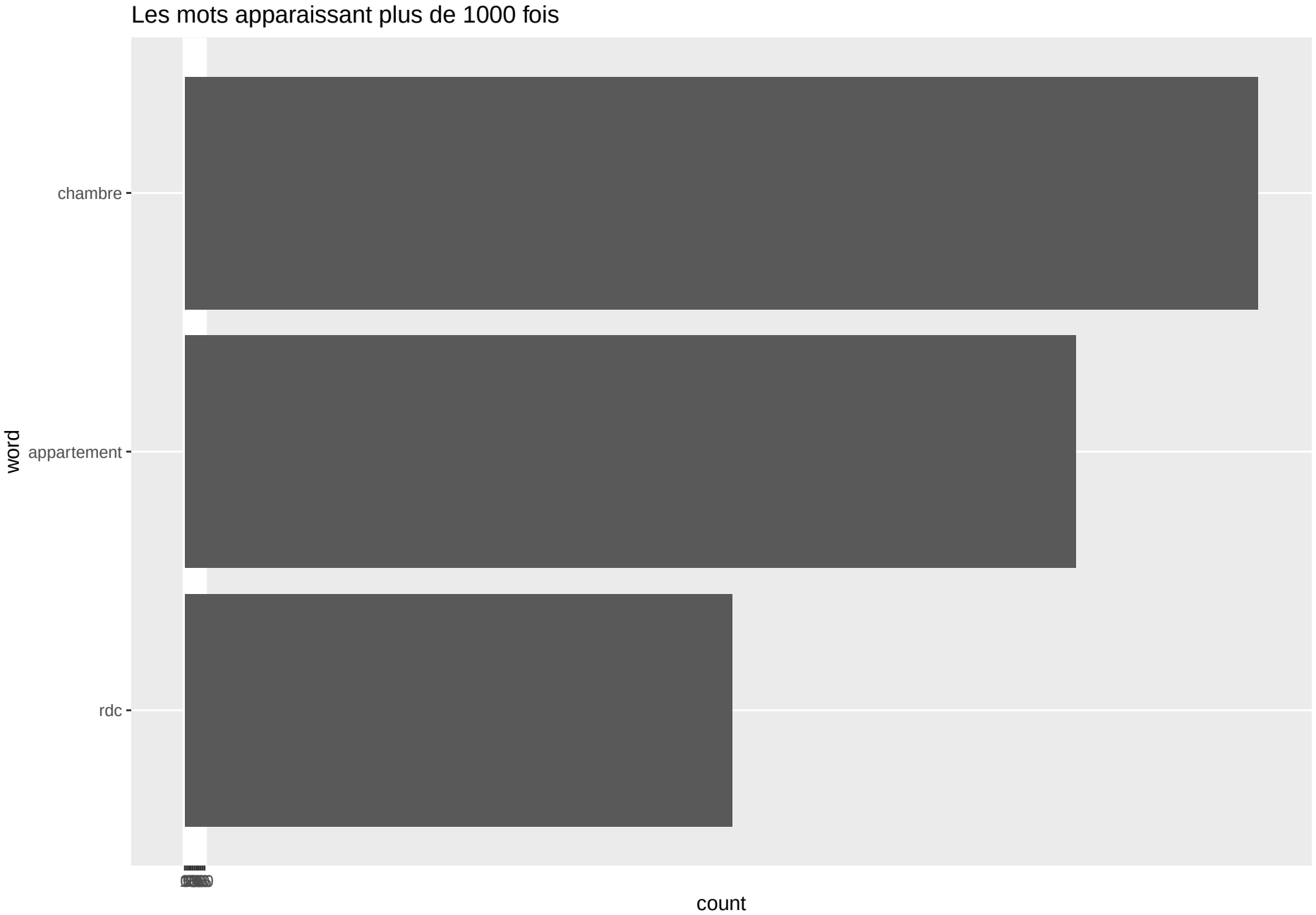


```
## [1] 511
```

De nombreux mots présents n'apportent aucune réelle valeur à notre analyse. Des mots comme **de**, **pour**, **les**, **le**, **la** sont ce qu'on appelle des mots vides (stop words).

Nous allons supprimer ces mots en utilisant la commande *anti_join(stop_words)*.

cas de la base rgph (Autre à préciser dans type de logement)

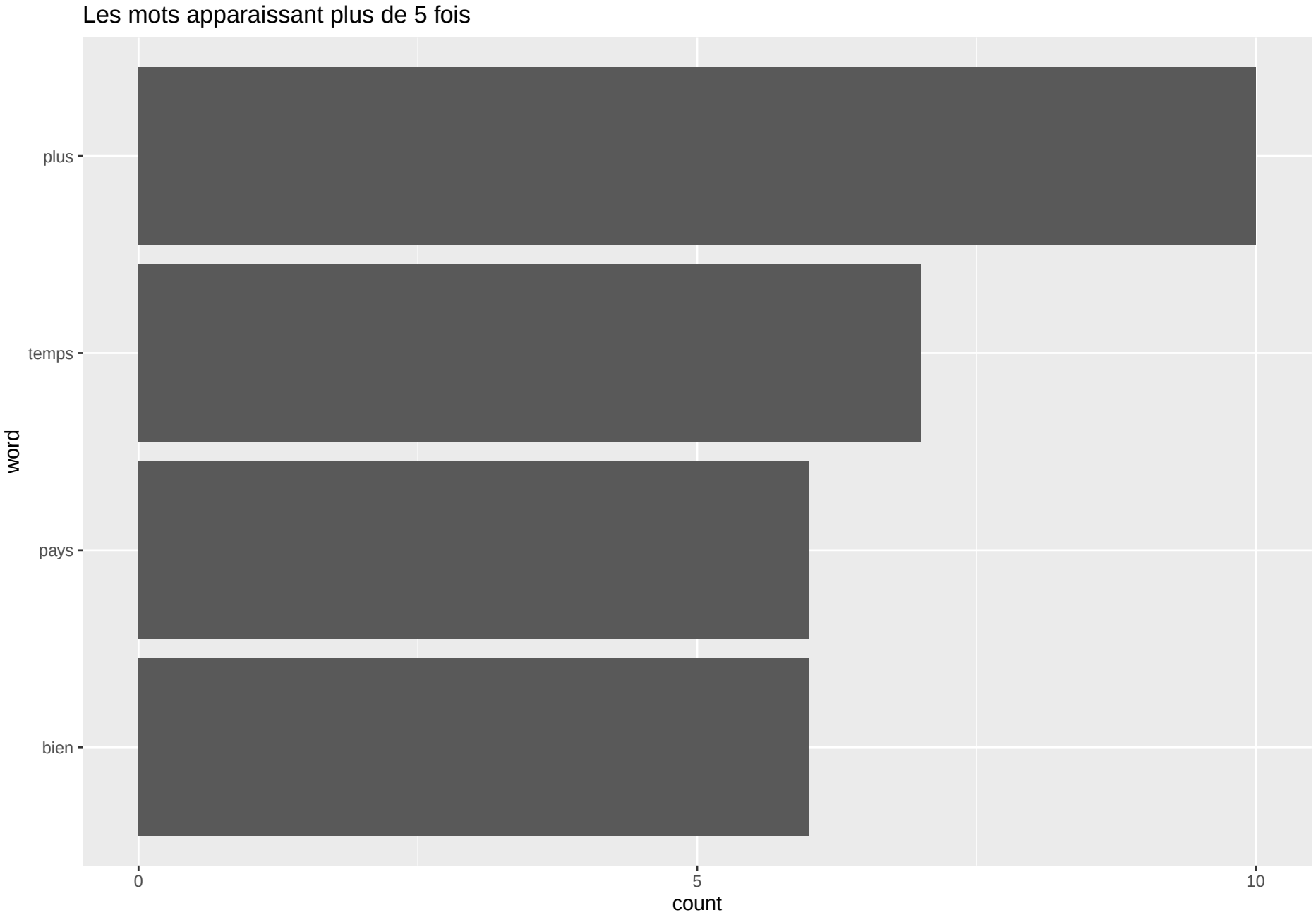

Les mots apparaissant plus de 1000 fois
chambre
appartement
rdc
word
count

Charger les stop words en français

```
## # A tibble: 10 x 1
##    word
##    <chr>
##  1 au
##  2 aux
##  3 avec
##  4 ce
##  5 ces
##  6 dans
##  7 de
##  8 des
##  9 du
## 10 elle
```

```
## [1] 317
```

On voit bien que le nombre de mots diminue suite à la supression des stop word. Comme nous pouvons le voir sur le graphique ci-dessous, il reste moins de mots, mais ils sont beaucoup plus pertinents pour l'analyse.

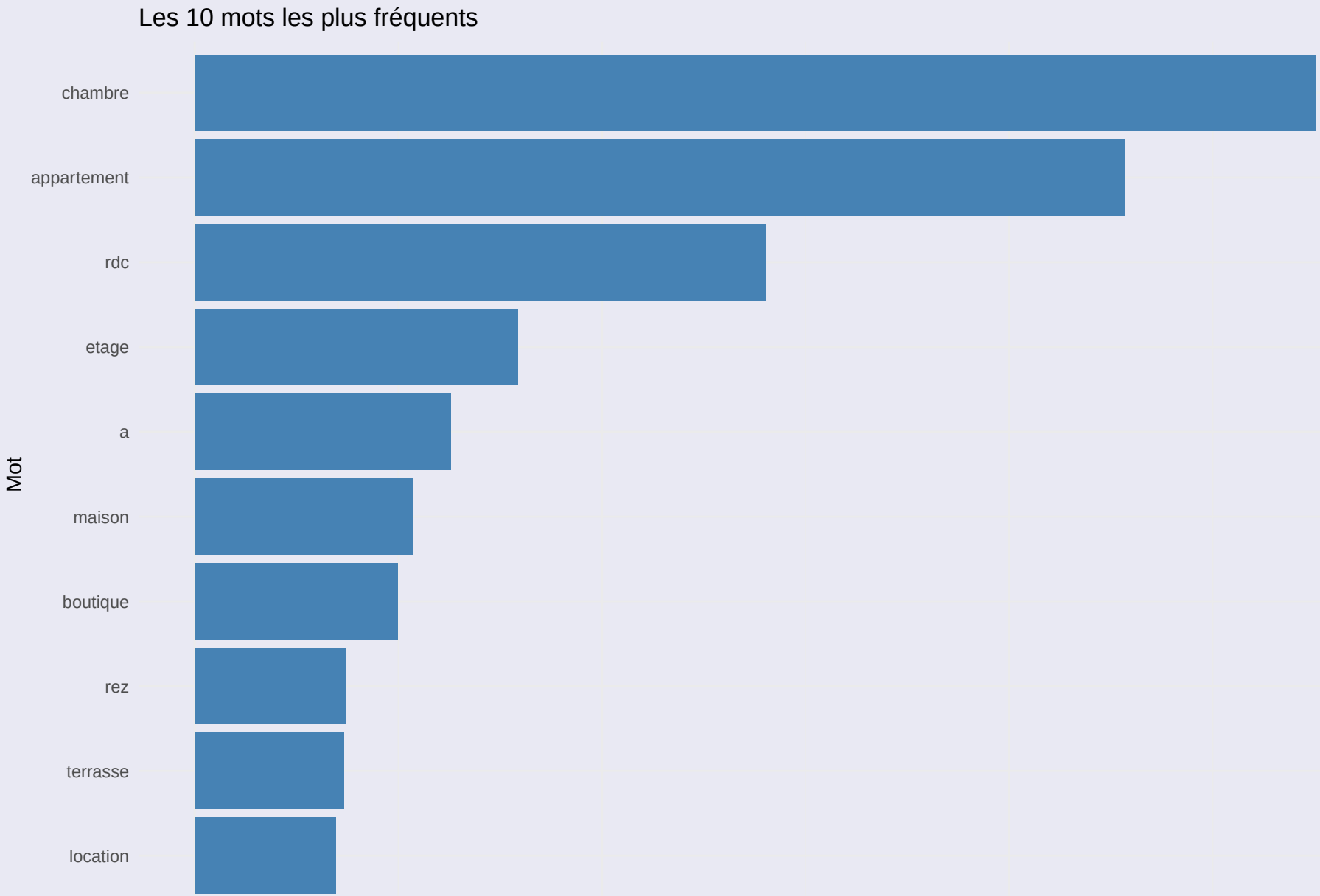

Les mots apparaissant plus de 5 fois
plus
temps
pays
bien
word
0
5
10
count


donner
pays
tout
faire
plus
bien
organiser
temps

L'application des stop word diminue le nombre de mots. Ceci le montre

```
## [1] 215
```

Cas du rgph5


Les 10 mots les plus fréquents
Mot
chambre
appartement
rdc
etage
a
maison
boutique
rez
terrasse
location

Racinisation

En racinisant, les mots *cultures* et *culture* par exemple se réduisent en *culture*. Voilà pourquoi le nombre total de mots diminue comme le résultat de cette commande l'illustre :

```
## [1] 201
```

Fréquence des mots avant le stemming

```
## # A tibble: 10 x 2
##    word          n
##    <chr>     <int>
##  1 plus         10
##  2 temps         7
##  3 bien          6
##  4 pays          6
##  5 donner        5
##  6 faire         5
##  7 organiser     5
##  8 tout          5
##  9 chaque        4
## 10 culture       4
```

Fréquence des mots après le stmming

```
## # A tibble: 10 x 2
```

On remarque que le stemming ne semble pas respecter la logique pour certains mots. En effet, la racinisation supprime les lettres *s* à la fin des mots comme *plus* et *temps*. Egalement le mot *paix* est réduit à *pai*. Cela constitut une limite majeure quant à la racinisation en langue française. C'est pourquoi dans ce qui suit, nous ferons fi de cette étape du prétraitement en utilisanat désormais seulement mes textes issus de l'application des stop word.

les analyse TF-IDF

Ci-dessous, nous voyons l'intégralité du tableau TF-IDF. Ce qui nous intéresse le plus, c'est la colonne tf_idf, car elle nous donne le classement pondéré ou l'importance des mots dans notre texte.

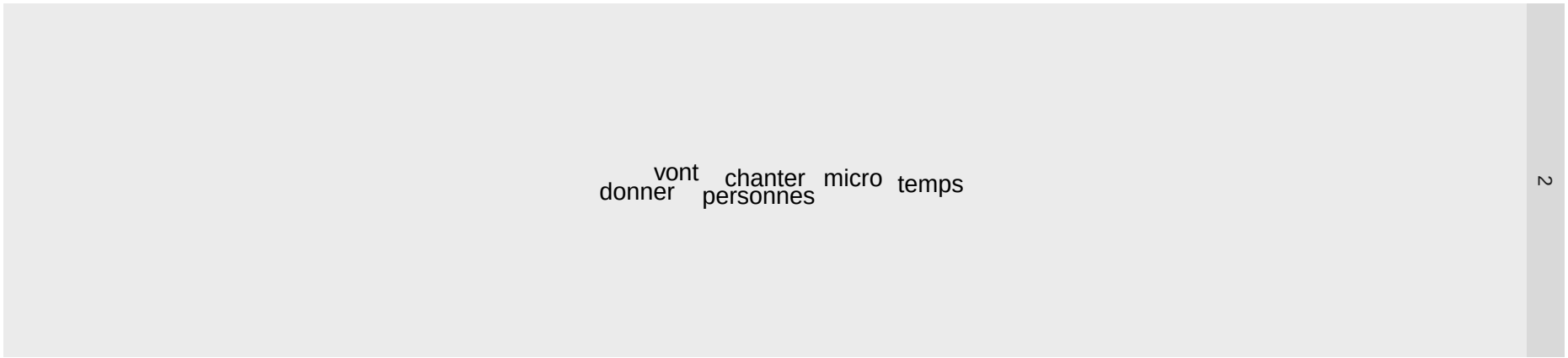
```
## # A tibble: 6 x 6
##   word          id count    tf   idf tf_idf
##   <chr>      <dbl> <int> <dbl> <dbl>  <dbl>
## 1 bien          65     2 0.25   2.20  0.549
## 2 cultures     117     2 0.105  2.71  0.285
## 3 dominante    102     2 0.25   3.81  0.952
## 4 pays          75     2 0.143  2.20  0.314
## 5 tout          32     2 0.222  2.42  0.538
## 6 a             39     1 0.2    3.81  0.761
```

Les simples décomptes de fréquences de mots peuvent être trompeurs et peu utiles pour bien comprendre nos données. Il est en fait intéressant de voir les mots les plus fréquents dans chaque texte.

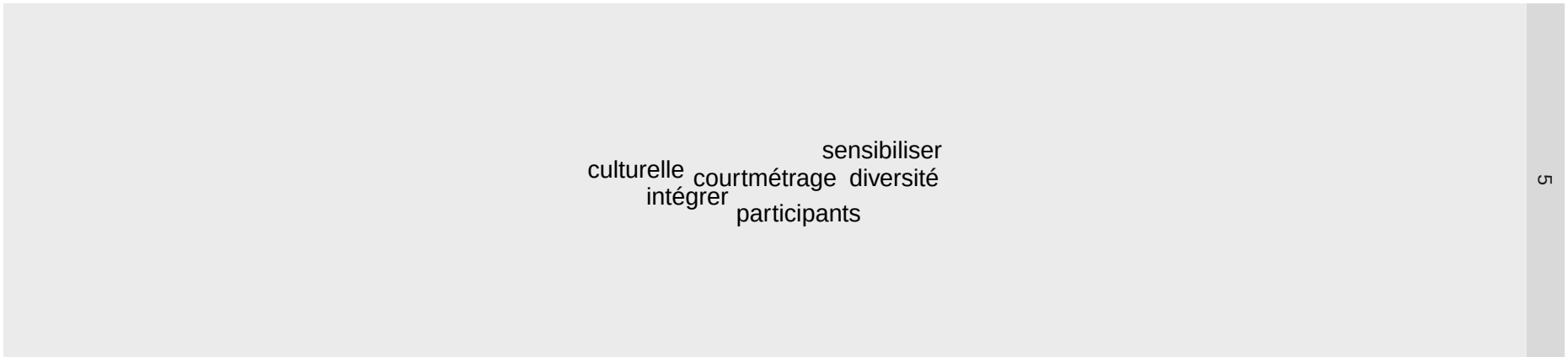
```
## # A tibble: 6 x 6
##   word          id count    tf   idf tf_idf
##   <chr>      <dbl> <int> <dbl> <dbl>  <dbl>
## 1 bien          65     2 0.25   2.20  0.549
## 2 cultures     117     2 0.105  2.71  0.285
## 3 dominante    102     2 0.25   3.81  0.952
## 4 pays          75     2 0.143  2.20  0.314
## 5 tout          32     2 0.222  2.42  0.538
## 6 a             39     1 0.2    3.81  0.761
```

Les simples décomptes de fréquences de mots peuvent être trompeurs et peu utiles pour bien comprendre nos données. Il est en fait intéressant de voir les mots les plus fréquents dans chaque texte.

On s'est limité au cinq premiers textes. Mais les textes correspondant aux identifiants 1 et 3 sont des NA et donc ont été isolés. Par ailleurs le résultat qui suit montre aussi qu'il ne faut pas se limiter à un simple dénombrement des textes mais à leur fréquence.


vont
donner
chanter
personnes
micro
temps
2


courts
puisse
entendre
qu'on
acteurs
micro
4


sensibiliser
culturelle
courtmétrage
diversité
intégrer
participants
5

Relations entre les mots

Jusqu'à présent, nous avons seulement examiné les mots individuellement. Mais que faire si nous voulons connaître les relations entre les mots dans un texte ? Cela peut être accompli grâce aux n-grammes.

```
## # A tibble: 6 x 2
##      id bigram
##   <dbl> <chr>
## 1     2 donner à
## 2     2 à temps
## 3     2 temps le
## 4     2 le micro
## 5     2 micro aux
## 6     2 aux personnes
```

Comme vous pouvez le voir dans le dataframe ci-dessus, certains bigrammes contiennent des mots vides (stop words) qui n'apportent pas beaucoup de valeur. Supprimons ces mots vides. Pour cela, nous allons d'abord séparer la colonne des bigrammes en deux colonnes distinctes nommées 'word1' et 'word2'. Ensuite, nous utiliserons deux fonctions de filtre pour supprimer les mots vides.

```
## # A tibble: 6 x 3
##      id word1     word2
##   <dbl> <chr>     <chr>
## 1     2 vont      chanter
## 2     4 sketchs   courts
## 3     4 bien      donner
## 4     4 acteurs   qu'on
## 5     4 qu'on     puisse
## 6     5 diversité culturelle
```

On peut maintenant compter les bigram et voir le résultat

```
## # A tibble: 6 x 3
##   word1       word2          n
##   <chr>       <chr>      <int>
## 1 chaque      pays           3
## 2 différentes cultures       2
## 3 donner      plus           2
## 4 faut        réduire        2
## 5 plus        grande         2
## 6 soirée      dansante       2
```

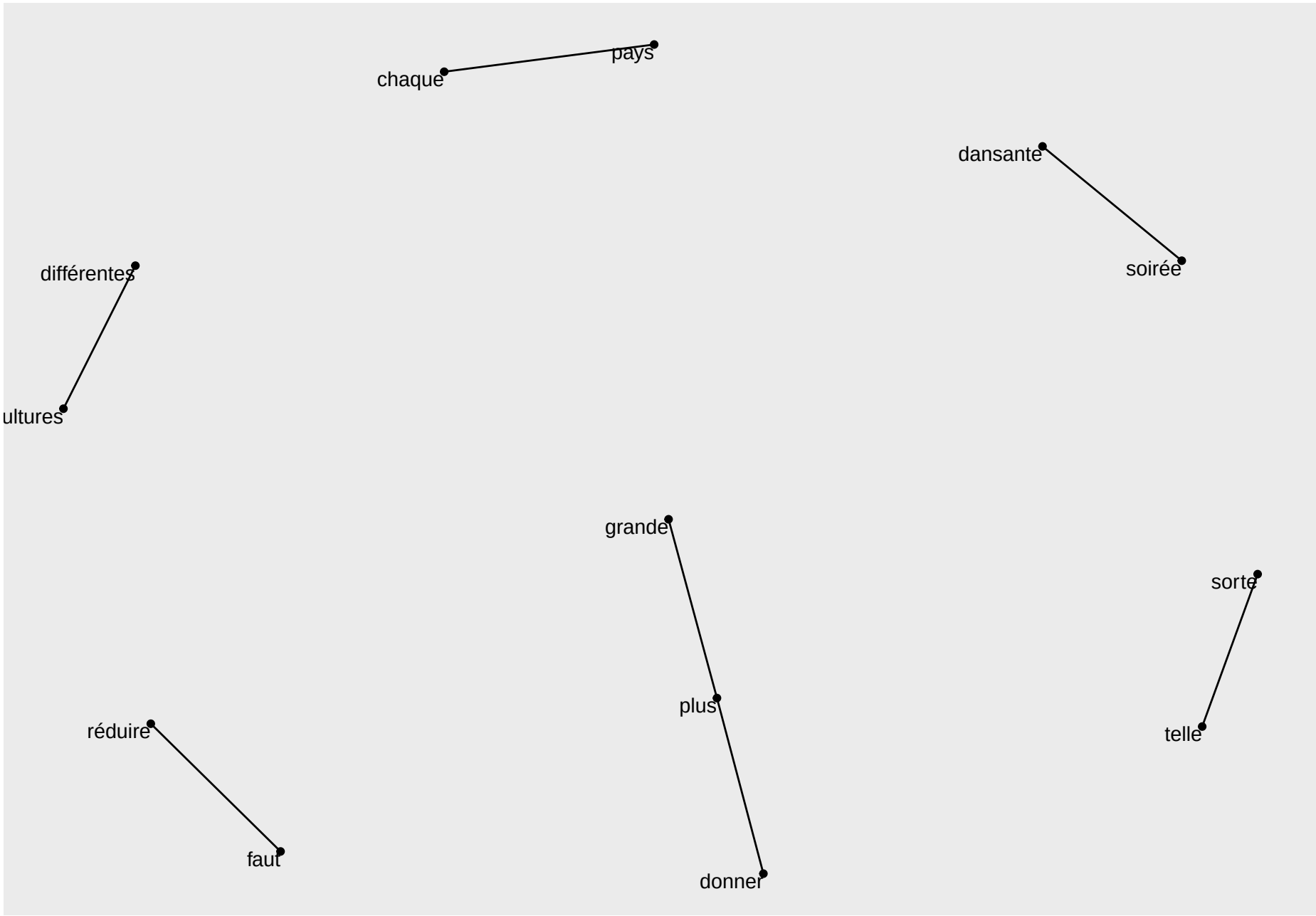
Comme précédemment, on peut aussi créer une mesure TF-IDF avec des n-grammes. Faisons-le maintenant.

```
## # A tibble: 6 x 6
## # Groups:   id [1]
##      id bigram              n    tf   idf tf_idf
##   <dbl> <chr>           <int> <dbl> <dbl>  <dbl>
## 1     2 aux personnes       1 0.111  3.81  0.423
## 2     2 donner à            1 0.111  3.81  0.423
## 3     2 temps le            1 0.111  3.81  0.423
## 4     2 vont chanter        1 0.111  3.81  0.423
## 5     2 à temps             1 0.111  3.81  0.423
## 6     2 personnes qui       1 0.111  3.11  0.346
```

Comme on peut le voir ci-dessus, beaucoup de valeurs TF-IDF sont identiques. Cela est en partie dû à la petite taille des textes.

Cas du rgph5

```
## # A tibble: 6 x 6
## # Groups:   id [4]
##   id    bigram          n    tf   idf tf_idf
##   <fct> <chr>       <int> <dbl> <dbl>  <dbl>
## 1 35    <NA>            1   1   NA      NA
## 2 65    <NA>            1   1   NA      NA
## 3 395   a herbe         1   0.5  7.98   3.99
## 4 395   maison a        1   0.5  5.18   2.59
## 5 397   en banco        1   0.5  6.88   3.44
## 6 397   maison en       1   0.5  4.48   2.24
```


chaque
pays
dansante
soirée
différentes
ultures
grande
plus
donner
sorte
telle
réduire
faut

Comme on peut le voir ci-dessus, de nombreux noms et d'autres informations ont été extraits des données.

Cas des trigrams

On peut aussi voir ci-dessous les trigrams

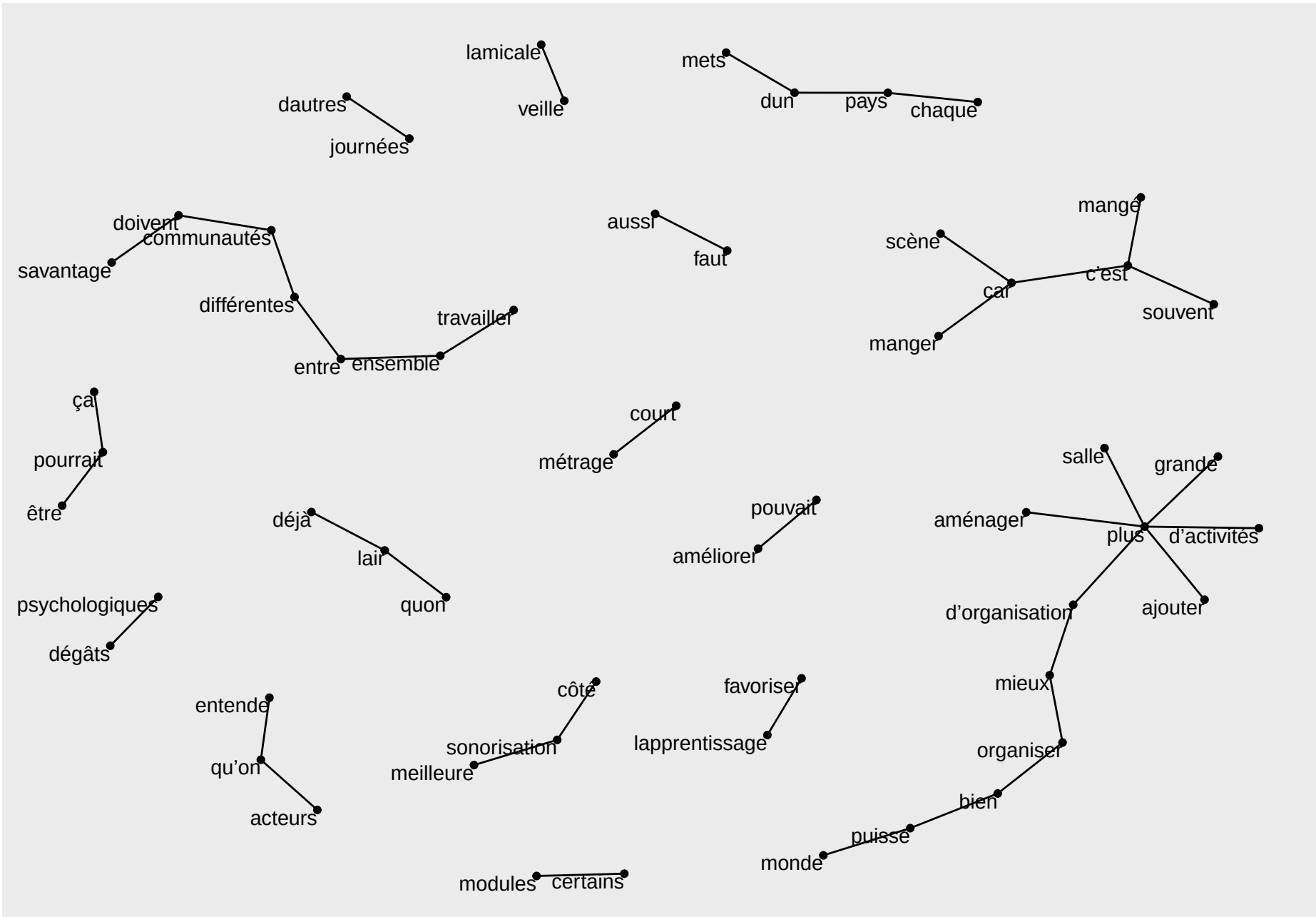
```
## # A tibble: 6 x 4
##      id word1         word2          word3
##   <dbl> <chr>         <chr>          <chr>
## 1     4 acteurs       qu'on          puisse
## 2     7 scène         car            habituellement
## 3    11 meilleure     sonorisation   côté
## 4    11 sonorisation  côté           technique
## 5    14 dautres       journées       continuellement
## 6    14 favoriser     lapprentissage culturel
```

```
## # A tibble: 6 x 4
##   word1    word2     word3            n
##   <chr>    <chr>     <chr>        <int>
## 1 acteurs  qu'on     puisse           1
## 2 ajouter  plus      d'activités      1
## 3 aménager plus      despace          1
## 4 bien     organiser l'événement      1
## 5 car      c'est     souvent          1
## 6 certains modules   statistiques     1
```

Comme précédemment, on peut aussi créer une mesure TF-IDF avec des trigrammes. Faisons-le maintenant.

```
## # A tibble: 6 x 6
## # Groups:   id [1]
##      id trigram                    n    tf   idf tf_idf
##   <dbl> <chr>                  <int> <dbl> <dbl>  <dbl>
## 1     2 aux personnes qui          1 0.125  3.81  0.476
## 2     2 donner à temps             1 0.125  3.81  0.476
## 3     2 micro aux personnes        1 0.125  3.81  0.476
## 4     2 personnes qui vont         1 0.125  3.81  0.476
## 5     2 qui vont chanter           1 0.125  3.81  0.476
## 6     2 temps le micro             1 0.125  3.81  0.476
```

Beaucoup de valeurs TF-IDF sont identiques. Cela est en partie dû à la petite taille des textes comme remarqué dans le cas bigram.


lamicale
veille
dautres
journées
mets
dun
pays
chaque
doivent
communautés
savantage
différentes
travailler
entre
ensemble
aussi
faut
mange
scène
c'est
car
souvent
manger
ça
pourrait
être
court
métrage
déjà
lair
quon
pouvait
améliorer
salle
grande
aménager
plus
d'activités
d'organisation
ajouter
psychologiques
dégâts
entende
qu'on
acteurs
côté
sonorisation
meilleure
favoriser
lapprentissage
mieux
organiser
bien
puisse
monde
modules
certains

Remarque

Normalement, on mettrait n $>$ 1 ou n $>$ 2 pour filtrer les trigrammes peu fréquents, mais dans notre cas, les textets sont très courts, donc les trigrammes se répètent très peu.

Dans notre base de données, le champ contenant les suggestions n'est pas obligatoire, ce qui signifie que plusieurs enregistrements présentent des valeurs manquantes (NA).

```
## [1] 128
```

```
## [1] 45
```

```
## [1] 83
```

3. Analyse Thématique (LDA)

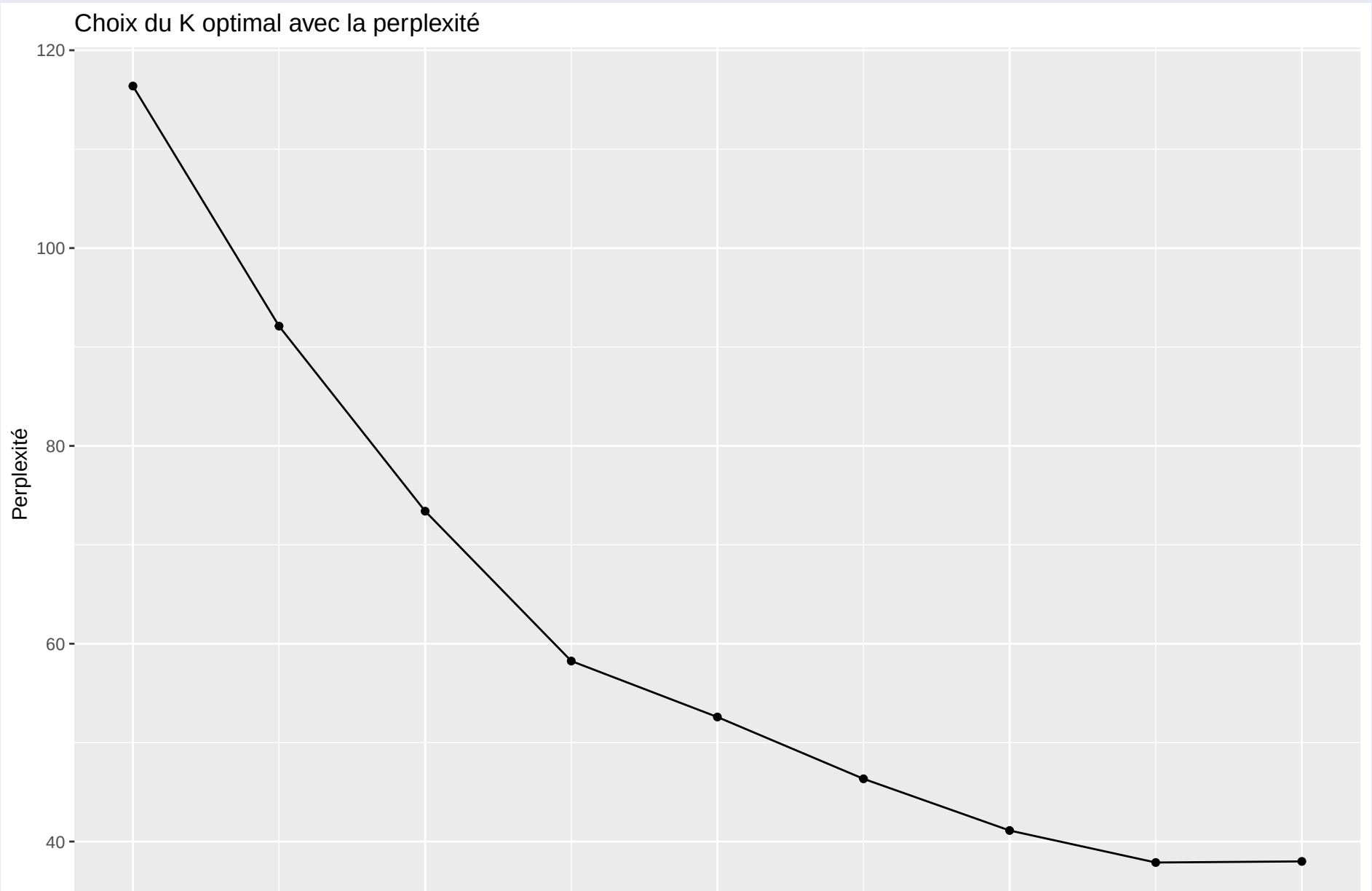
La méthode LDA (Latent Dirichlet Allocation). LDA repose sur deux grands principes : Chaque document est un mélange de plusieurs sujets. Chaque sujet est un mélange de mots

Un exemple classique serait de supposer qu'il existe deux grands sujets dans les actualités : la politique et le divertissement. Le sujet politique contiendra des mots comme élu, gouvernement, tandis que le sujet divertissement contiendra des mots comme film, acteur. Mais certains mots peuvent apparaître dans les deux, comme prix ou budget. LDA va identifier : les mélanges de mots qui composent chaque sujet, et les mélanges de sujets qui composent chaque document.

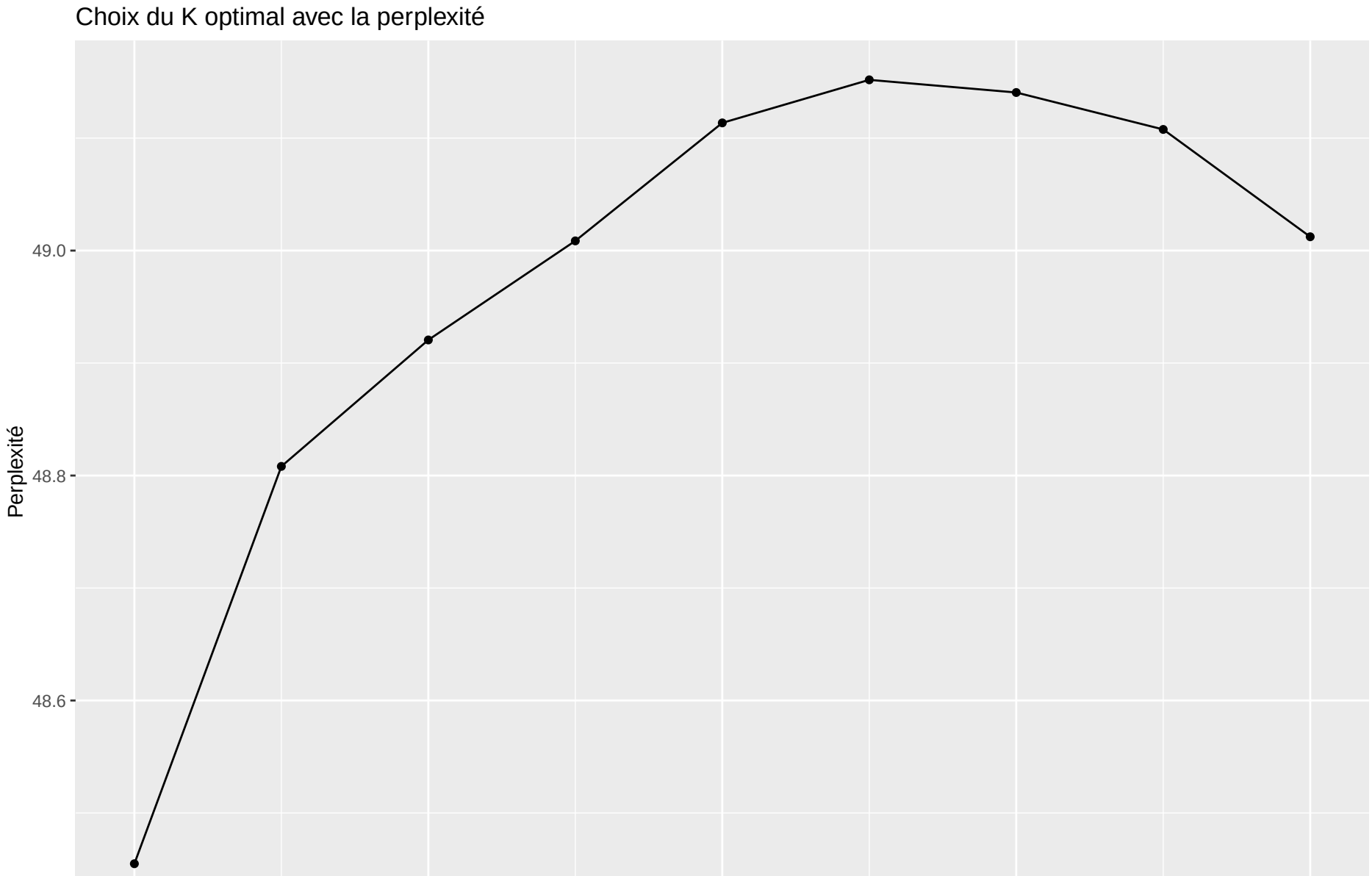
Choix du nombre k de thèmes

Dans le cadre de la modélisation thématique avec LDA (Latent Dirichlet Allocation), un des éléments clés du paramétrage est le choix du nombre de thèmes (K). Ce paramètre n'est pas déterminé automatiquement par le modèle ; il doit être choisi par l'utilisateur, en fonction des données et des objectifs de l'analyse. Or, le nombre de thèmes a un impact direct sur la qualité et la lisibilité du modèle.

Application de la perpelexité pour le choix de K



Cas du rgph5


Choix du K optimal avec la perplexité
Perplexité
49.0
48.8
48.6

Cas où le graphe de la perpeléxité ne produit pas un résultat escompté (comme celui du rgph5)

Limites : pas trop flexible surtout en cas de grands volumes de données

Une autre solution (validation humaine)

```
tokenized_textes1 %>% count(word, sort = TRUE) %>% rename(count = n) %>% filter(count > 1000) %>% mutate(word = reorder(word, count)) %>% ggplot(aes(x = count, y = word)) + geom_col() + labs(title = "Les mots apparaissant plus de 1000 fois") + scale_x_continuous(breaks = seq(0, 50, 5))
```

- Isolement des termes récurrents à partir de la visualisation
- Regroupement des mots de même sens
- Poursuivre le processus pour avoir une base composés uniquement des mots isolés (ceux que l'on gère bien)

lancement du modèle

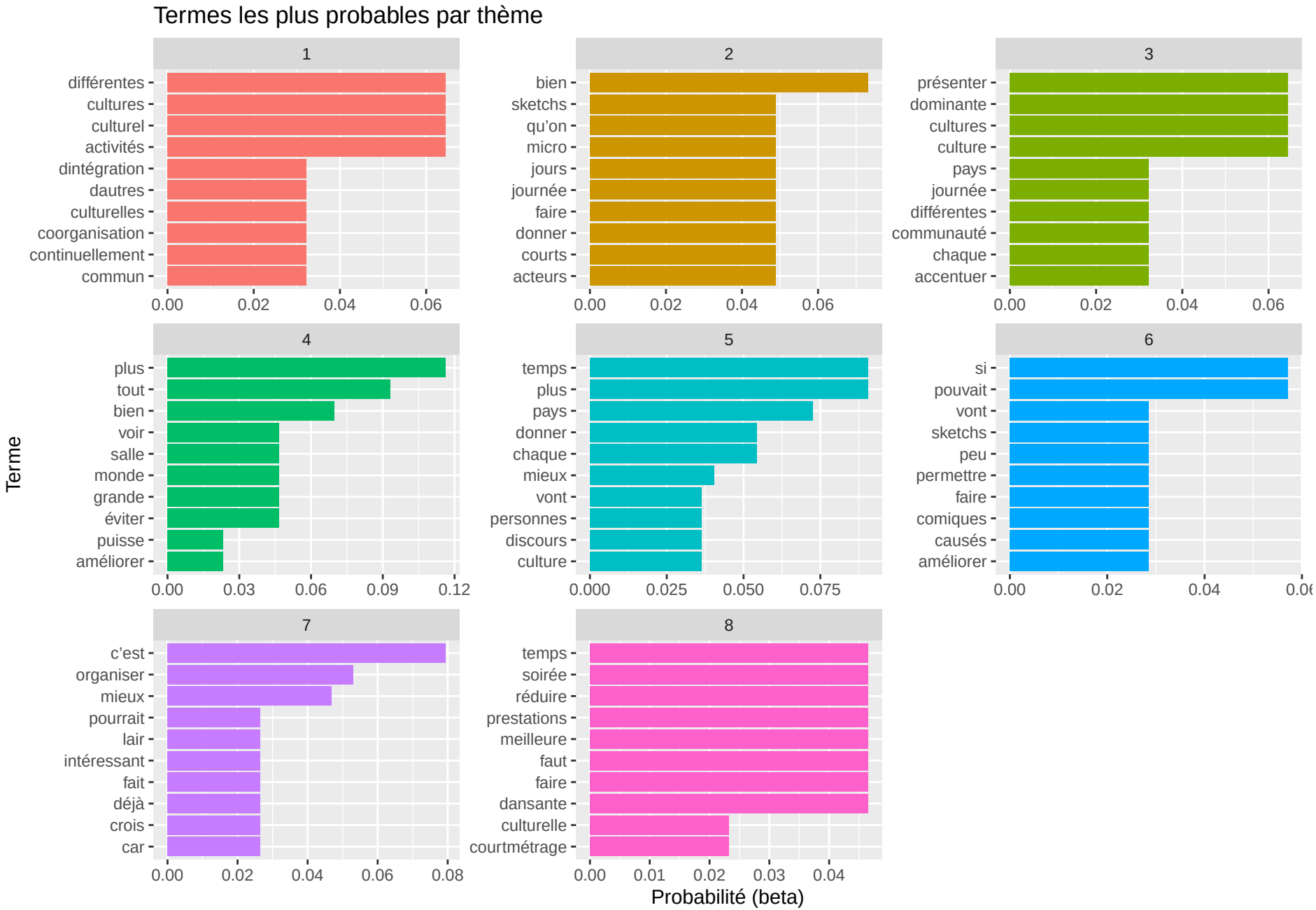
```
lda_model <- LDA(df_dtm, k = 8, control = list(seed = 1234))
# Termes par thème
terms_by_topic <- tidy(lda_model, matrix = "beta")
```

```
## # A tibble: 1,720 x 3
##    topic term          beta
##    <int> <chr>        <dbl>
##  1     1 chanter 5.02e-154
##  2     2 chanter 4.94e-324
##  3     3 chanter 5.02e-154
##  4     4 chanter 1.88e-154
##  5     5 chanter 1.81e-  2
##  6     6 chanter 3.48e-154
##  7     7 chanter 2.77e-154
##  8     8 chanter 1.88e-154
##  9     1 donner  1.14e-153
## 10     2 donner  4.88e-  2
## # i 1,710 more rows
```

La colonne beta représente la probabilité qu'un mot donné appartienne à un thème particulier.

```
## # A tibble: 8 x 2
##   topic     n
##   <int> <int>
## 1     1    10
## 2     2    10
## 3     3    10
## 4     4    10
## 5     5    10
## 6     6    10
## 7     7    10
## 8     8    10
```

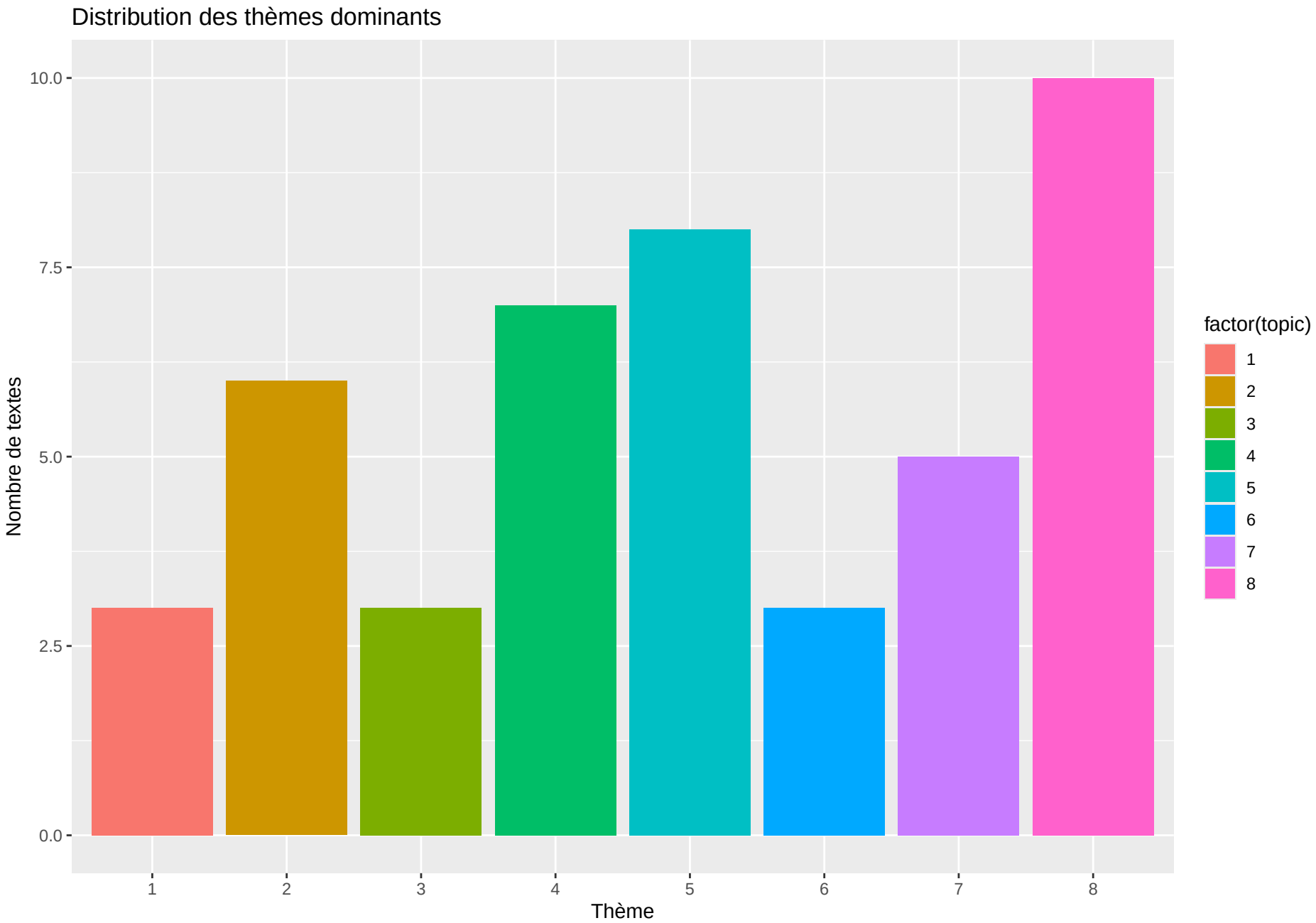
Analyse en Composantes Principales (ACP)



Ici pour chaque texte, on peut voir la probabilité qu'il a d'appartenir à chacun des thèmes.

```
## # A tibble: 8 x 2
##   topic     n
##   <int> <int>
## 1     1     3
## 2     2     6
## 3     3     3
## 4     4     7
## 5     5     8
## 6     6     3
## 7     7     5
## 8     8    10
```

Pour chaque thème, on voit le nombre de textes


Distribution des thèmes dominants
Nombre de textes
10.0
7.5
5.0
2.5
0.0
1
2
3
4
5
6
7
8
Thème
factor(topic)
1
2
3
4
5
6
7
8

Labellisation (très subjective)

```
# Labelliser les thèmes

textes_gamma <- textes_gamma %>%
  mutate(topic_label = case_when(
    topic == 1 ~ "Diversité culturelle et activités communes",
    topic == 2 ~ "Performances artistiques et intervention scé
    topic == 3 ~ "Présentation des cultures par les communauté
    topic == 4 ~ "Aménagement de l'espace et la gestion du tem
    topic == 5 ~ "Organisation",
    topic == 6 ~ "Suggestions d'amélioration",
    topic == 7 ~ "Organisation générale et impression globale"
    topic == 8 ~ "animation",
    TRUE ~ "Autre"
  ))
```

4. Approche Alternative avec BERTopic

BERTopic est un outil puissant de topic modeling (modélisation de sujets) qui permet d'extraire automatiquement des thèmes principaux à partir de textes non structurés. Il se distingue des approches classiques comme LDA par sa capacité à capturer des relations sémantiques en se basant sur le texte et non des motstokenisés.

Configuration pour travail en python

Importation des modules Python

Chaque module que nous importons est lié à une fonctionnalité clé :

- sentence_transformers : gestion des modèles d'embedding de texte
- hdbscan : algorithme de clustering utilisé par BERTopic
- bertopic : la librairie principale pour la modélisation de sujets
- umap : utilisé pour projeter les embeddings dans un espace de plus faible dimension

```r
sentence_transformers <- import("sentence_transformers")
hdbscan <- import("hdbscan")
bertopic <- import("bertopic")
umap <- import("umap")  # important pour fixer le random_state
```

Ici on utilise ‘paraphrase-MiniLM-L6-v2’, un modèle rapide et efficace. Mais il existe d’autres variétés plus puissantes mais qui sont plus robustes en mémoire Le tableau qui suit donne quelques détails.

Table 1: Tableau comparatif de modèles d'embedding utilisables avec BERTopic

Modele	Taille	Precision_Semantique
paraphrase-distilbert-base-nli-stsb	768	Bonne précision sémantique
bert-base-nli-mean-tokens	768	Très bonne précision sémantique
all-mpnet-base-v2	768	Excellente précision sémantique

```r
# Préparation des données

docs <- Texte_JI_filtered$Texte
ids <- Texte_JI_filtered$id  # on garde l'id associé à chaque

result <- topic_model$fit_transform(docs)


# Extraction des résultats
topics <- result[[1]]
probs <- result[[2]]
```

```
##  [1] 4 3 5 3 1 0 0 0 2 2 1 0 2 2 3 0 5 0 3 4 1 1 1 1 3 3 1
## [39] 1 5 4 0 1 0 4
```

```
##  [1] 0.7649678 1.0000000 1.0000000 0.7850124 1.0000000 0.87
##  [8] 1.0000000 1.0000000 1.0000000 0.9240548 1.0000000 0.87
## [15] 1.0000000 1.0000000 1.0000000 0.9533261 1.0000000 1.00
```

```r
#  Reconstruction du data.frame avec id + texte + classe
base_categorisee <- data.frame(
  id = ids,
  texte = docs,
  classe = topics,
  proba = probs
)

# Affichage des infos sur les thèmes trouvés
topic_info <- topic_model$get_topic_info()
```

```
##   Topic Count                                  Name
## 1     0    15                      0_de_la_pays_les
## 2     1    10                  1_bien_pour_soit_son
## 3     2     6              2_tout_une_soirée_monde
## 4     3     6        3_sketchs_des_courts_acteurs
## 5     4     5     4_temps_prestations_réduire_le
## 6     5     3 5_court_métrage_culturelle_intégrer
##
## 1                                  de, la, pays, les, préser
## 2                                          bien, pour,
## 3                                        tout, une, soi
## 4                                          sketchs, de
## 5                                  temps, prestations, réduire
## 6 court, métrage, culturelle, intégrer, créativité, preuve,
##
## 1 Que chaque pays présente sa culture de telle sorte les pe
## 2
## 3
```

5. Catégorisation

Dans cette section, nous tentons de catégoriser les textes en se basant sur les diff'rents thèmes générés par le modèle.

```
topic_info$label <- c(
  "Célébration et partage des cultures nationales",
  "Aspect technique et organisation",
  "Mieux aménager l'espace",
  "Sketchs et prestations",
  "Gestion du timing lors des interventions",
  "Touche créative et courmétrages"
)
```

```r
base_categorisee <- merge(
  base_categorisee,
  topic_info[, c("Topic", "label")],
  by.x = "classe",
  by.y = "Topic",
  all.x = TRUE
)
base_categorisee <- subset(base_categorisee, select = -c(class

doc_vides <- data.frame(
  id = Id_texte_NA
)

# 2. Identifier les noms des autres colonnes (sauf "document"
autres_colonnes <- setdiff(names(base_categorisee), "id")

# 3. Ajouter des NA pour les autres colonnes
doc_vides[autres_colonnes] <- NA
```

```r
# 4. Fusionner avec la base existante
textes_by_topic_complet <- rbind(base_categorisee, doc_vides)

# 5. Optionnel: trier par document si nécessaire
textes_by_topic_complet <- textes_by_topic_complet[order(texte

# Joindre les thèmes dominants avec les tweets originaux

Texte_JI$Texte <- NULL

textes_classified <- Texte_JI %>%
  inner_join(textes_by_topic_complet, by = c("id" = "id"))
```

```
## # A tibble: 6 x 5
##      id ‘Classe de l’étudiant :‘ ‘Nationalité de l’étudiant
##   <dbl> <chr>                    <chr>
## 1     1 AS2                      Congo
## 2     2 ISEP1                    Cameroun
## 3     3 AS2                      Congo
## 4     4 AS1                      Sénégal
## 5     5 AS1                      Cameroun
## 6     6 ISE1 Eco                 Sénégal
```

Table 2: Suggestions pour améliorer l'organisation de la journée d'intégration

id	texte	label
1	NA	NA
2	Donner à temps le micro aux personnes qui vont chanter.	Gestion du timing lors des interventions
3	NA	NA
4	Faire des sketchs courts et bien donner le micro aux acteurs.	Sketchs et prestations
5	Intégrer un court-métrage sur la diversité culturelle pour sensibiliser.	Touche créative et courmétrages
6	NA	NA
7	Faire des sketch courts et donner le micro aux acteurs sur la scène.	Sketchs et prestations
8	NA	NA
9	Améliorer le son pour que tout soit bien audible.	Aspect technique et organisation

CONCLUSION

L'analyse textuelle nécessite un prétraitement rigoureux, mais les outils sont souvent mieux optimisés pour l'anglais, limitant leur efficacité pour le français. Des méthodes comme LDA (basée sur les co-occurrences de mots) ont des limites sémantiques, tandis que BERTopic, utilisant des embeddings contextuels (comme BERT), capture mieux le sens des textes. Cependant, l'analyse automatique reste imparfaite face à la diversité linguistique et nécessite une validation humaine pour des résultats fiables.

CONCLUSION

A retenir

Prétraitement crucial mais biaisé vers l'anglais.
LDA $\rightarrow$ limite sémantique ;
BERTopic $\rightarrow$ meilleur sens contextuel.
Analyse textuelle = utile mais à valider par l'humain.