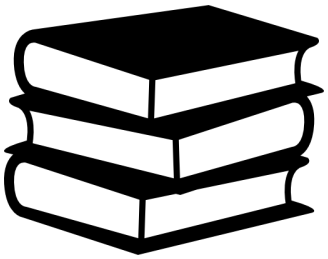
Projet Analyse Multivariée sur R

Frédéric Francine

Table des matières



1 Introduction

Peut on grouper des livres par leur titre ?

Voilà la question à laquelle on va essayer de répondre. En effet, il existe différents algorithmes de clustering, donc il est légitime d'essayer de les appliquer sur tout type de donnée. Toutefois, on se heurte à un problème de représentation des données. Pour nous, il est facile de classer des livres ensembles par genre mais la machine n'applique les méthodes de clustering que sur un format spécifique. Selon la représentation choisie, on n'a plus la donnée de départ et dès lors on perd en information, ce qui est problématique dans nos calculs de distance.

Pour répondre à notre question, on va utiliser les techniques de NLP. De cette manière, on travaille directement sur les mots, donc on ne perd pas d'information à première vue. De plus, il existe des algorithmes pour grouper ces ensembles de mot. On appelle cela du **topic modeling**, car on appelle les groupes de documents formés sont appelés **topic**. On rajoute les définitions suivantes qui sera utile pour la compréhension du sujet. Un **document** est un ensemble de mot et un **corpus** est un ensemble de documents. Ici le corpus sera l'ensemble des titres des livres de notre jeu de donnée et le titre du livre représentera un document. Une fois que le corpus réalisé, on réalise une matrice **Document-Terme**(*Document Term Matrix* en anglais), qui comporte en ligne les documents du corpus et en colonne, les termes de l'ensemble du corpus. Par exemple, notons A , la matrice document terme, le coefficient A_{ij} correspond à une fonction de i et de j, par exemple on peut avoir la fonction qui retourne le nombre de fois que j apparaît dans le document i. Cette matrice est très grande, d'où il est vital de bien épurer le corpus.

Enfin, une fois que cette matrice crée, on utilise principalement la **LDA (Latent Dirichlet Allocation)**,une technique pour récupérer nos topic, mais nous verrons aussi l'utilisation de **bigram** et aussi de la **STM**, une autre technique qui nous permet de récupérer des critères sur la formation des topic.

2 Travail sur les données

On utilisera les packages suivants :

```
library(gutenbergr)
library(tm)
library(SnowballC)
library(wordcloud)
library(dplyr)
library(stringr)
library(textstem)
```

Le package gutenbergr est une base de données sur l'ensemble des textes du domaine publique, on peut retrouver notamment la constitution des États-Unis. On obtient l'ensemble des livres anglais, par la commande gutenberg_work() auquel, on va garder uniquement la colonne titre. On traite les données manquantes.

```
df_title = data.frame(gutenberg_works()$title)

sum(is.na(df_title))

df_title = na.omit(df_title)
colnames(df_title) = "title"
```

2.1 La longueur des titres

On fait un histogramme sur la longueur des mots, pour voir la longueur des documents dans le jeu de données.

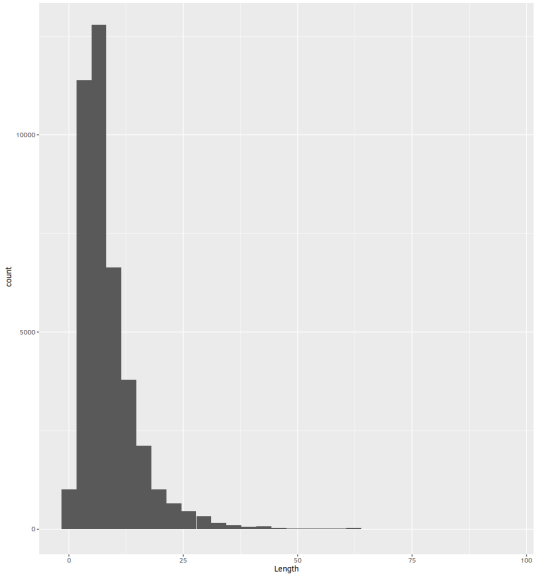


FIGURE 1 – Histogramme sur la longeur des documents

On voit une forte concentration des documents entre 0 et 10. Regardons cette zone de plus près.

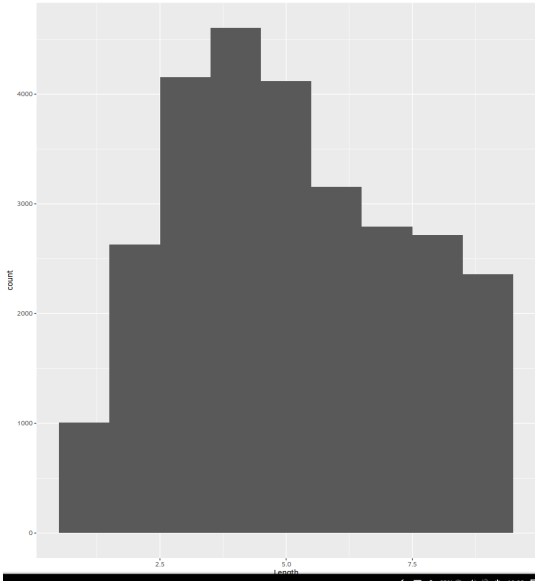


FIGURE 2 – Histogramme pour les documents inférieur à 10 mots

On voit que le titre avec 4 mots est le plus fréquent, suivi de près des titres de 5 à 3 mots. Toutefois, cela constitue des documents courts, difficile de faire du topic modeling dessus. C'est pourquoi, on décide de prendre uniquement les documents supérieurs à 7 mots.

```
len = len = df_title %>% mutate(title=str_count(title,"\\w+"))
ggplot(len,aes(x=title)) + geom_histogram()

len2 = filter(len,str_count(Length,"\\w+")<10)
ggplot(len2,aes(x=title)) + geom_histogram(bins=9)

df_title = filter(df_title,str_count(title,"\\w+")>7)
dim(df_title)
```

On obtient 18273 titres sur les 40737 de la base. Ensuite, on crée un corpus par les fonctions du package tm et qu'on épure. Le schéma classique en text-mining est de mettre le texte en minuscule, d'enlever les mots non pertinents, de retirer la ponctuation et les espaces blancs. Toutefois, l'analyse qui suit, m'a fait adapté à l'analyse. En construisant la matrice Document Terme, on peut savoir les termes qui reviennent le plus souvent par la commande find_Freq_Terms().

2.2 Corriger les titres

```
> findFreqTerms(title_dtm,1000)
[1] "book"    "life"    "story"   "stories" "history" "volume"  "—"
[8] "vol"
```

FIGURE 3 – Les mots les plus fréquents

On voit que le terme - revient le plus souvent alors qu'il ne représente rien. Il n'a pas été éliminé du removePunctuation, car il ne s'agit pas du tiret du clavier. Une fois éliminé par removeWords, on constate un autre problème. Les documents contenant ces tirets forment des mots composés.

```
 [1] "united"          "bible"               "james"
 [4] "king"            "version"             "adventures"
 [7] "world"           "book"                "papers"
[10] "american"        "life"                "complete"
[13] "plays"           "story"               "essays"
[16] "house"           "modern"              "tale"
[19] "days"            "tales"               "poems"
[22] "woman"           "john"                "stories"
[25] "little"          "rise"                "short"
[28] "history"         "volume"              "copyright"
[31] "letters"         "years"               "golden"
[34] "last"            "three"               "literature"
[37] "lady"            "england"             "romance"
[40] "great"           "prince"              "richard"
[43] "works"           "love"                "samuel"
[46] "sketches"        "memoirs"             "union"
[49] "henry"           "diary"               "quixote"
[52] "trail"           "mirror"              "revision"
[55] "english"         "madame"              "addresses"
[58] "volumepart"      "part"                "douayrheims"
[61] "instruction"     "state"               "amusement"
[64] "pepysvolume"     "novelsfontainevolume" "cleopatravolume"
[67] "emperorvolume"   "caxtonsfamily"       "picturevolume"
[70] "novelvolume"     "parisiansvolume"     "challoner"
[73] "punchinello"
```

FIGURE 4 – Titres collés

Par exemple, ici nous avons *emperorvolume* ou *novelvolume* alors qu'ils ne sont pas attachés. On utilise alors la fonction du package stringr str_replace_all pour remplacer le caractère par un espace blanc. De même, en filtrant le dataframe, on a vu la présence des caractères de saut de lignes \n et \r mais aussi d'apostrophe \'s, on applique la même fonction que précédemment.

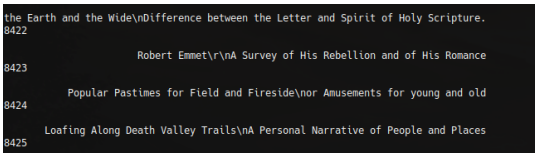


FIGURE 5 – Titres avec \r et \n

Enfin le dernier problème rencontré est la présence de mot au pluriel et au singulier, tel que *story* et *stories*. On peut les rassembler en utilisant le package textstem, c'est ce qu'on appelle la **lemmatization**. On décide de supprimer les mots volume et complete car on les trouve peu pertinent. On laisse par contre d'autres mots tel que book ou novel car ils peuvent représenter un certain type de livre. Finalement, on a le programme suivant.

```
mywords= c("volume","complete")
f = function(x) str_replace_all(x,"([\r\ n  ])|(‘s)"," ")
oldw = getOption("warn")
options(warn=-1)
corp_title = corp_title %>%
tm_map(content_transformer(tolower)) %>%
tm_map(content_transformer(f)) %>%
tm_map(removeWords,c(mywords,stopwords("en"))) %>%
tm_map(removePunctuation) %>%
tm_map(content_transformer(lemmatize_strings)) %>%
tm_map(stripWhitespace)
options(warn=oldw)
```

On fait ensuite la matrice Document terme.

```
title_dtm = DocumentTermMatrix(corp_title)
inspect(title_dtm)
```

```
> inspect(title_dtm)
<<DocumentTermMatrix (documents: 18273, terms: 17956)>>
Non-/sparse entries: 133698/327976290
Sparsity           : 100%
Maximal term length: 25
Weighting          : term frequency (tf)
Sample             :
       Terms
Docs    american book boy history life london new story tale work
  10971        0    1   0       0    0      0   0     0    0    0
  17667        1    0   0       1    1      0   0     0    0    1
  17814        0    0   0       0    0      0   0     0    0    0
  3883         0    0   0       2    0      0   0     1    0    0
  4204         0    0   0       0    0      0   0     0    0    0
  4234         0    0   0       1    0      0   0     0    0    0
  6349         0    0   0       0    0      0   0     0    0    0
  8880         0    0   0       0    0      0   0     0    0    0
  9223         2    0   0       0    0      0   0     0    0    0
  9969         0    0   0       0    0      0   0     0    0    0
```

FIGURE 6 – Une partie de la matrice Document-Terme

On a un échantillon ici de la matrice document terme. On a bien en colonnes, les termes et en lignes les documents. L'intersection ligne/colonne indique le nombre d'occurence du mot dans le document. On s'attend malheureusement à une matrice remplie de 0, c'est ce qu'on peut remarquer par la valeur de **sparsité** et la proportion de coefficients de matrice qui sont nons nuls dans l'ensemble des coefficients de la matrice.

De manière similaire que la commande find___Freq_Term(), on peut faire un wordcloud pour visualiser les mots qui apparaissent le plus dans notre corpus.

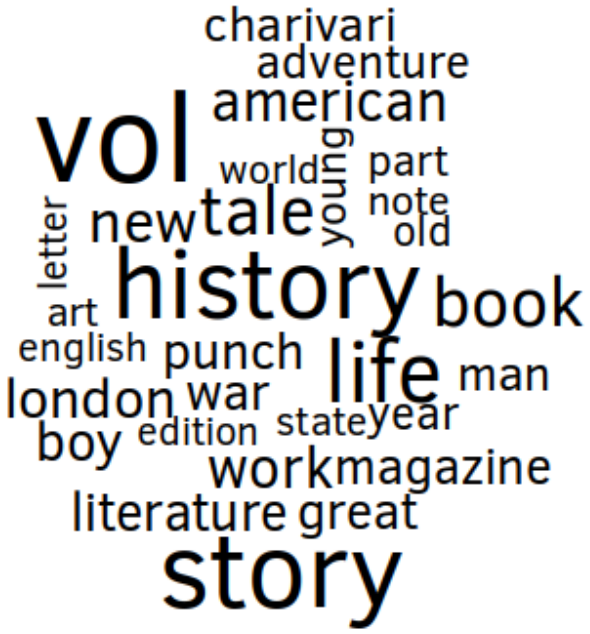


FIGURE 7 – Wordcloud sur le corpus

Il n'est pas surprenant de retrouver des mots liés au champs lexical de la littérature. En observant de plus près, le mot *punch*, on découvre qu'il est associé au mot *London* pour désigner un journal anglais du XIXème siècle.

On peut aussi changer la mesure de comptage, celle décrite dans l'introduction est la mesure Fréquence-Document mais il est courant d'utiliser la mesure *Tf-Idf*, car elle retiendrait des mots pertinents, mais pour l'algorithme qu'on utilise dans le topic modeling, on a besoin de la matrice en Fréquence-Document.

3 Stratégie de topic modeling

3.1 Première approche : LDA

Avant d'appliquer l'algorithme LDA, on doit vérifier que la matrice comporte bien des documents non vides, c'est à dire des un document avec aucun terme de la matrice Document-Terme. On réalise l'opération suivante :

```
ind = unique(dtm$i)
dtm=dtm[ind,]
```

Le modèle **LDA** est un modèle probabiliste, souvent utilisé en topic modeling. On utilise le théorème de Bayes et des distributions de Dirichlet et Multinomial, pour avoir une probabilité

a posteriori sur l'appartenance d'un document à un topic. L'avantage, est qu'on n'a pas besoin de fournir les topics, on les extrait automatiquement du corpus, ce qui est bien pratique quand on ne connait pas l'ensemble des documents. Nous allons reprendre le schéma et l'explication du site scikit-learn [1] pour expliquer l'algorithme.

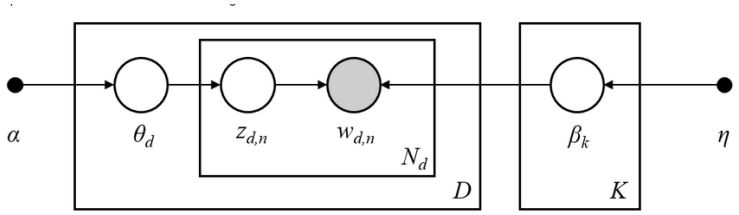


FIGURE 8 – Le modèle probabiste LDA

Le schéma représente une itération de l'algorithme LDA. Ici D représente l'ensemble des documents et K représente l'ensemble des topics, tandis que β_k suivent $\text{Dir}(\eta)$ et que les variables θ_d suivent $\text{Dir}(\alpha)$. On a l'algorithme suivant :

1. Pour chaque topic k dans K, on tire β_k selon $\text{Dir}(\eta)$
2. Pour chaque document d dans D, on tire θ_d selon $\text{Dir}(\alpha)$
3. Pour chaque mot i dans d :
   - On tire un assignement à un topic selon une $\text{Multinomial}(\theta_d)$
   - On tire un mot selon une $\text{Multinomial}(\beta_{z_i d})$

La probabilité a posteriori s'écrit :

$$p(z, \theta, \beta | w, \alpha, \eta) = \frac{p(z, \theta, \beta | \alpha, \eta)}{p(w | \alpha, \eta)}$$

Le principal hyperparamètre de l'algorithme à déterminer est le nombre de topic, comme dans les techniques de clustering. Il y a une technique qui ressemble à la méthode du coude. En effet, on veut minimiser une mesure appelé la perplexité, et on prends la perplexité la plus basse.

```
find_best_nb =
function(nb_topic){
  mod = LDA(dtm,k=nb_topic,method="Gibbs",
  control=list(alpha=0.5, iter=1000, seed=12345, thin=1))
  c(nb_topic,perplexity(mod,dtm))
}

graph = sapply(2:12,find_best_nb)
graph = data.frame(t(graph))

ggplot(data=graph,aes(x=X1,y=X2))+
      geom_line()+
      geom_point()+
      labs(title="Evolution de la perplexite selon le nombre de topic",
      x="Nombre de topic",
      y="Perplexite")+
      theme_minimal()
```

1. Explication du modèle du site scikit-learn.

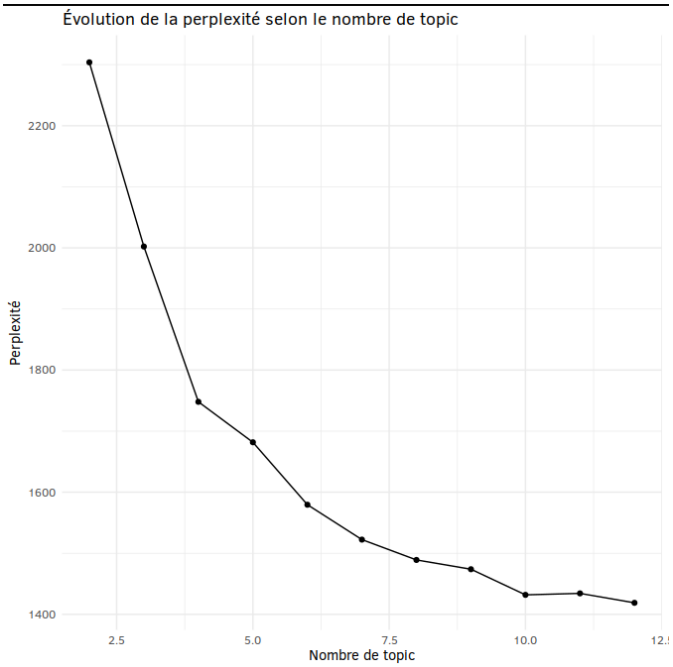


FIGURE 9 – La méthode du coude revisitée

Le graphique n'a pas l'air très pertinent. On peut utiliser le package ldatuning qui applique des méthodes spécifiques de minimisation ou de maximisation d'un critère. On choisit les critères minimisant et on obtient le graphe suivant.

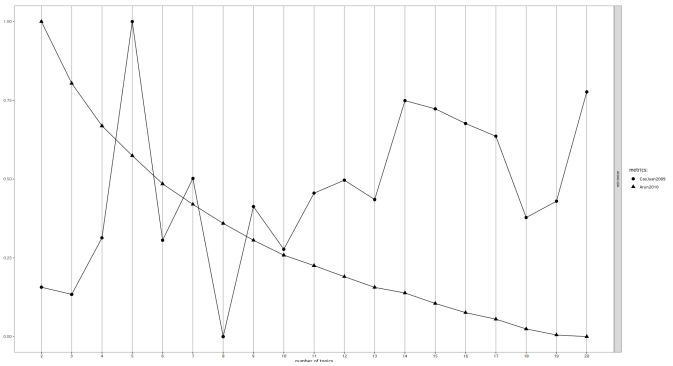


FIGURE 10 – Algorithmes determinant le meilleur nombre de topic

Le graphe selon l'algorithme de *Arun2010* suit une trajectoire similaire à la courbe obtenu par le coude. La courbe atteint son minimum pour 20 topics. Tandis que le graphe avec *CaoJuan2009* n'est pas monotone et il atteint son minimum pour 8 topics.

Appliquons LDA avec un nombre de topic valant 8.

On regarde les probabilités a posteriori des mots pour chaque topic.

```
1 library(tidytext)
2
```

```
top =
function(mod){
  beta_mat = tidy(mod,matrix="beta")

  top_word = beta_mat %>%
  group_by(topic) %>%
  top_n(10,beta) %>%
  ungroup() %>%
  arrange(topic,-beta)
  top_word %>%
  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()
}

top(lda_8)
```

FIGURE 11 – Les probabilités posteriori des termes avec 8 topics

Certains topics ne sont pas représentatif d'un genre comme le premier

```
        1         2         3         4         5         6         7         8
0.1136364 0.1136364 0.1136364 0.1136364 0.1681818 0.1500000 0.1136364 0.1136364
```

FIGURE 12 – Les 10 mots les plus représentatifs des 8 topics

Comme attendu de notre coup d'oeil sur les mots caractérisant les topics, on voit que les topics ne discrimine pas le document. On doit augmenter le nombre de topic pour avoir une meilleur discrimination. On prend 20 en accord selon le deuxième algorithme. On obtient les probabilités a posteriori suivantes pour les termes. On a les probabilités posteriori des termes selon chaque topic.

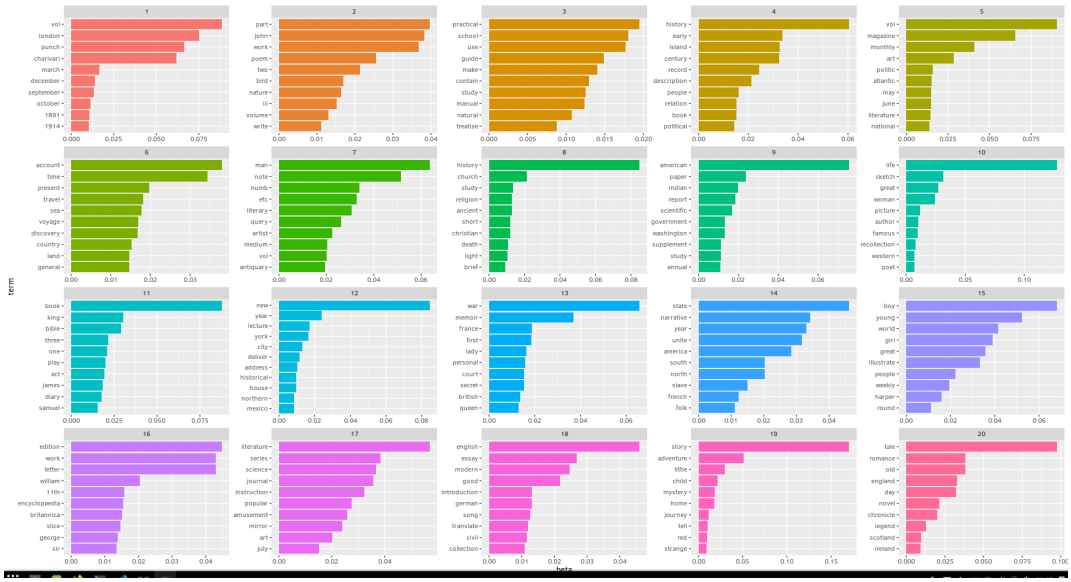


FIGURE 13 – Les 10 mots les plus représentatifs des 20 topics

On peut toujours se demander de la pertinence même avec 20 topics. En effet, le mot *king* du topic 11, aurait été plus pertinent dans le topic 13, qui contient le mot *queen*, mais aussi les termes *France*, *British* et *court* qui suggèrent un topic sur l'histoire et les monarchies. De même, que les topic 9 et 14 semblent être sur les États-Unis, difficile de les différencier. On va observer les probabilités posteriori du premier document

```
         1          2          3          4          5          6          7
0.04545455 0.04545455 0.04545455 0.04545455 0.04545455 0.04545455 0.04545455
         8          9         10         11         12         13         14
0.04545455 0.04545455 0.04545455 0.04545455 0.06363636 0.04545455 0.11818182
        15         16         17         18         19         20
0.04545455 0.04545455 0.04545455 0.04545455 0.04545455 0.04545455
```

FIGURE 14 – Les probabilités posteriori pour le premier document avec 20 topics

Les probabilités a posteriori ne sont pas hautes, mais elles discriminent puisqu'à part la probabilité du topic 14 qui est à environ 0.1 mais les autres probabilités sont inférieurs à 0.1. La *Constitution des États-Unis* est classé dans le topic 14 et les mots les plus fréquents de ce topic semblent liés à l'histoire des États-Unis.

3.2 Tester une STM

On va maintenant appliquer une STM(**Structural Topic Model**). L'algorithme STM a un principe similaire à une LDA, sauf qu'il prend en compte les **meta-data**. . D'où l'idée de faire du topic modeling à l'aide d'une structure sous-ajacente. On pourrait utiliser en meta-data, les noms des auteurs. Toutefois, on s'éloigne de notre problématique de départ. La raison pourquoi nous allons faire une STM sans meta-data est que le package lié à la STM sur R, fournit des critères pour la pertinence des topics

```
library(stm)
cor = readCorpus(dtm,type="slam")
out = prepDocuments(documents=cor$documents,vocab=cor$vocab)

stm.search = searchK(documents = out$documents,
vocab = out$vocab,
K = 10:30,
init.type = "Spectral")
plot(stm.search)

mod_stm = stm(documents = out$documents,vocab=out$vocab,K=13,init.type="Spectral",
    seed=831)
plot(mod_stm,n=4,text.cex=.8)

#matrice des probabilit s a posteriori des documents selon les topic
mod_stm$theta[1,]
```

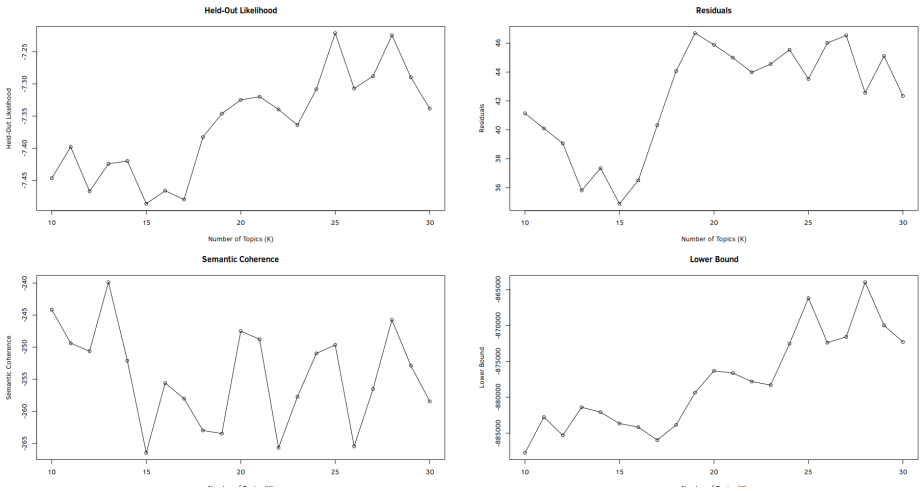


FIGURE 15 – Recherche du nombre de topic

Le critère principal a observer est la **cohérence**. Comme son nom l'indique, on regarde si les topics ne sont pas trop aléatoires et qu'il y a un sens de regrouper les documents à ce topic. Ici le maximum est atteint pour 13 topics. De plus les résidus sont faibles pour 13 topics, donc on va faire une STM avec 13 topics.

On peut aussi voir les termes les plus réprésentatifs des topics.

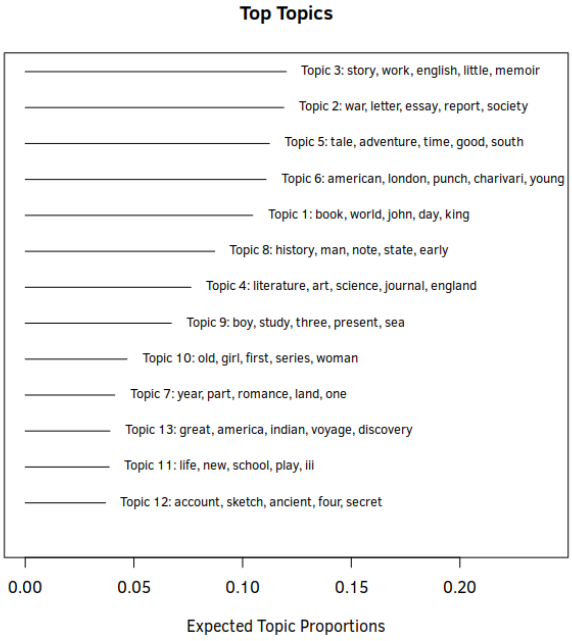


FIGURE 16 – Les 4 mots les plus représentatifs des 13 topics

Maintenant, on regarde l'assignation des topics.

```
> mod1$theta[1,]
 [1] 0.05594844 0.12724576 0.04819333 0.15094254 0.05244690 0.06204265
 [7] 0.04044630 0.26934216 0.04157813 0.03420946 0.03619561 0.03556939
[13] 0.04583933
```

FIGURE 17 – Les probabilités posteriori pour le premier document avec 13 topics

La constitution américaine est liée à 3 topics principalement, en particulier le topic 8. On a des probabilités plus hautes qu'en LDA, mais on n'a pas un topic qui discrimine le plus.

3.3 L'utilisation de bigram

Jusqu'à présent, on avait utilisé uniquement les termes, mais il serat judicieux de les grouper. Par exemple le mot *United*, peut référer à un livre des États-Unis mais aussi à un livre sur l'esprit collectif alors que le groupe de mot *United States* serait plus utile pour grouper les livres en lien avec les États-Unis. Le fait de considérer un groupe de deux mots est appelé **bigram**. On peut aussi constituer un groupe de plus de 2 mots. Un groupe de n mots est appelé **n-gram**

En utilisant les bigram, on augmente le nombre de termes avec le même nombre de document. On utilise le package quanteda, pour construire une nouvelle matrice Document-Terme.

```
library(quanteda)
corp_qtd = corpus(corp_title)
n_gram = tokens_ngrams(quanteda::tokens(corp_qtd),n=1:3)
dtm_qtd = dfm(n_gram)
nfeat(dtm_qtd)
```

On a 192742 termes, on utilise ensuite l'algorithme LDA avec 8 et 20 topics et on regarde si l'assignation à un topic est meilleure.

```
lda_8_qtd = LDA(dtm_qtd,8,method="Gibbs")
top(lda_8_qtd)

lda_20_qtd = LDA(dtm_qtd,20,method="Gibbs")
top(lda_20)
```

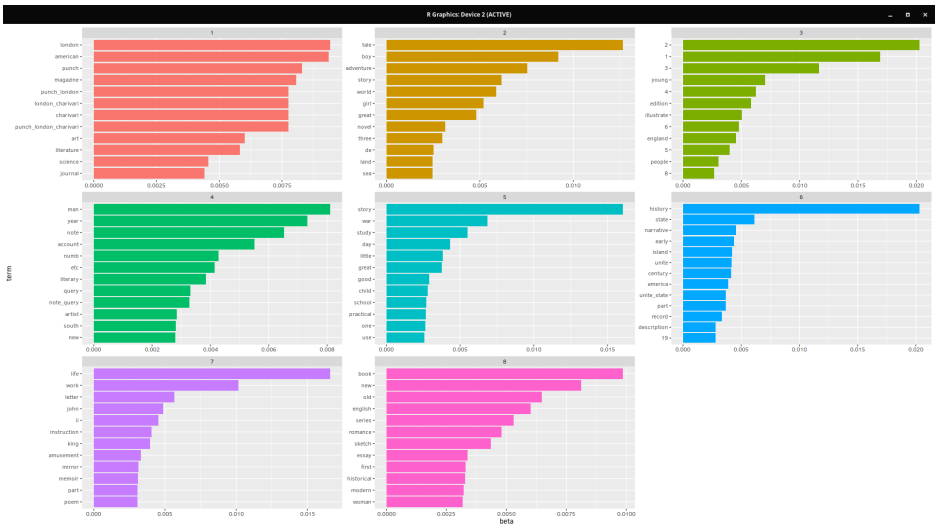


FIGURE 18 – Les 10 mots les plus représentatifs des 8 topics

Il y a peu de bigram dans les mots les plus communs de chaque topic. On retrouve *unite_state* dans le topic 6 qui correspond aux États-Unis et au journal *Punch, or London Charivari*. Par contre, le topic 3 n'est pas pertinent, car il comporte plus des chiffres que des mots. Ici, on aurait du retirer les nombres.

```
      1         2         3         4         5         6         7         8
0.1008065 0.1008065 0.1169355 0.1491935 0.1008065 0.2137097 0.1008065 0.1169355
```

FIGURE 19 – Les probabilités posteriori pour le premier document avec 8 topics

On a les résultats similaires qu'avec la première LDA, donc 8 reste un mauvais nombre de topic même avec les bigram.

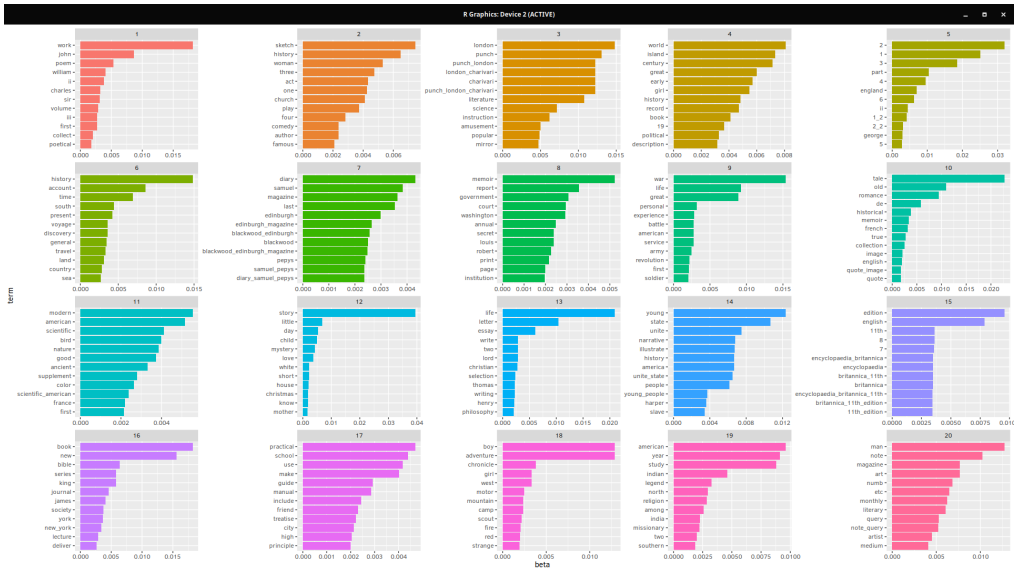


FIGURE 20 – Les 10 mots les plus représentatifs des 20 topics

```
       1        2        3        4        5        6        7
0.04032258 0.04032258 0.04032258 0.04032258 0.05645161 0.04032258 0.04032258
       8        9       10       11       12       13       14
0.04032258 0.04032258 0.04032258 0.04032258 0.04032258 0.04032258 0.15322581
      15       16       17       18       19       20
0.10483871 0.04032258 0.04032258 0.04032258 0.04032258 0.04032258
```

FIGURE 21 – Les probabilités posteriori pour le premier document avec 20 topics

Ici la constitution américaine est un mélange du topic 14 et 15, contrairement à l'assignation à un seul topic de la première LDA. Toutefois, ce n'est pas si grave, comme puisque les deux topics font un genre historique vu leurs mots.

4 Conclusion et ouverture

Le veritable problème rencontré est le fait que nos documents sont très courts malgré notre sélection. En effet, le topic modeling a du mal à bien fonctionner pour des documents courts, d'où c'est un problème. Le deuxième problème rencontré est la non reproducibilité de la LDA. En effet, il n'est pas possible malheureusement de retrouver les topic présentés dans ce document, une version avec seed de la LDA n'est pas encore point dans R.

Par ailleurs, il existe d'autres algorithme pour faire du topic modeling. Dans le package topicmodels, il y a l'algorithme CTM mais il a été écarté de l'étude par sa lenteur. De même, il existait des algorithmes "*algébriques*" tel que le LSA et NMF, qui consiste à décomposer la matrice document terme en un produit de matrice. Toutefois, R n'arrivait pas à factoriser la matrice document terme, par manque de puissance. Par ailleurs, utiliser le word2vec aurait donné une meilleure analyse, puisqu'avec un modèle entrainé, on aurait eu rassemblés les mots entre eux par le sens, mais je n'arrivais pas à utiliser keras.

Enfin, il est bien connu qu'on ne doit pas juger un livre à sa couverture, en particulier à son titre. On peut grouper deux livres ensembles par des similarités sur le titre, mais le contenu peut différer, donc même si on pouvait grouper des titres, elles ne seraient parfaites.