

# **NLP avec R pour les sciences sociales**

Sophie Balech

28/05/2024

# Table of contents

<b>Introduction</b>	<b>5</b>
Ressources en ligne . . . . .	5
 <b>I Niveau 1 : Initiation</b>	 <b>6</b>
<b>1 Prise en main</b>	<b>7</b>
1.1 Introduction . . . . .	7
1.2 Premières manipulations . . . . .	7
1.3 Charger des données . . . . .	7
1.3.1 Un tableau de données . . . . .	7
1.3.2 Une collection de fichier textes . . . . .	8
1.3.3 Reconnaissance Optique des Caractères (OCR) . . . . .	8
 <b>2 Pré-traitements</b>	 <b>11</b>
2.1 Introduction . . . . .	11
2.2 Les données . . . . .	11
2.3 Premières analyses des données . . . . .	13
 <b>3 Création du corpus et premières observations du corpus</b>	 <b>20</b>
 <b>4 Nettoyage du corpus</b>	 <b>24</b>
 <b>5 Filtrer le corpus des termes trop fréquents</b>	 <b>30</b>
 <b>6 La loi de Zipf</b>	 <b>34</b>
 <b>7 Mesures de fréquence</b>	 <b>35</b>
7.1 Pondération tf-idf . . . . .	35
 <b>8 Comprendre le sens des termes</b>	 <b>40</b>
 <b>9 Analyse du sentiment</b>	 <b>42</b>
 <b>10 Les données</b>	 <b>43</b>

<b>11</b>	<b><i>Sentiment analysis</i></b>	<b>44</b>
11.1	Les sentiments . . . . .	46
11.2	Les émotions . . . . .	49
<b>12</b>	<b>Évolution du corpus dans le temps</b>	<b>52</b>
<b>13</b>	<b>Nuage de mots comparés</b>	<b>55</b>
13.1	En fonction des années . . . . .	55
13.2	En fonction des sentiments . . . . .	56
<b>II</b>	<b>Niveau 2 : Approfondissements</b>	<b>58</b>
<b>14</b>	<b>Annotations et dépendances syntaxiques</b>	<b>59</b>
<b>15</b>	<b>Les données</b>	<b>60</b>
<b>16</b>	<b>Les traitements préliminaires</b>	<b>62</b>
<b>17</b>	<b>Co-occurrences</b>	<b>63</b>
17.1	Représentation en réseau des termes co-occurents . . . . .	67
<b>18</b>	<b>Annotations</b>	<b>69</b>
18.1	Détecter les langues . . . . .	69
18.2	POS . . . . .	69
<b>19</b>	<b>Les dépendances syntaxiques</b>	<b>76</b>
<b>20</b>	<b><i>Topic Analysis</i></b>	<b>78</b>
<b>21</b>	<b>Les données</b>	<b>79</b>
<b>22</b>	<b><i>Topics Analysis</i></b>	<b>80</b>
22.1	Le modèle LDA avec topicmodels . . . . .	81
22.2	<i>Topic Analysis</i> à partir de l'annotation des <i>part of speech</i> . . . . .	82
22.3	Déterminer le nombre de topics optimal . . . . .	84
22.4	Représentation graphique . . . . .	87
<b>23</b>	<b><i>Theory-Driven LDA</i></b>	<b>89</b>
<b>24</b>	<b>Expliquer les notes</b>	<b>93</b>
24.1	NPS . . . . .	93
24.2	En fonction des topics . . . . .	99

<b>Appendices</b>	<b>102</b>
<b>A Webscraping</b>	<b>102</b>
<b>References</b>	<b>107</b>

# Introduction

Ceci est le support pour le cours “NLP avec R pour les sciences sociales” réalisé auprès des doctorants de l’école doctorale SHS de l’UPJV.

L’objectif est de donner aux apprenants des méthodes et outils pour traiter de larges corpus de texte pour répondre à leur problématique de recherche.

L’environnement de traitement des données est R et Rstudio.

[Ici](#), une présentation en guise d’introduction.

## Ressources en ligne

R et RStudio

- <https://www.r-project.org/>
- <https://rstudio.com/products/rstudio/>

Des ressources en ligne

- [Introduction à R et au tidyverse](#)
- [R for data science](#)
- [Hands-On Programming with R](#)
- [Text mining with R](#)
- [Tutoriels Quanteda](#)
- [Les techniques du NLP pour la recherche en sciences de gestion](#)
- [NLP avec r et en français - un Manuel synthétique](#)
- [Quarto pour communiquer](#)
- Le séminaire du Collège de France : [Apprendre les langues aux machines - B. Sagot](#)

## **Part I**

### **Niveau 1 : Initiation**

# 1 Prise en main

## 1.1 Introduction

Le document de travail contient deux types d'éléments : du texte pour expliquer et présenter ce que l'on fait et du code pour réaliser les manipulations de données, les analyses et les graphiques. On commence toujours un document avec un bloc de code de setup, pour lister et charger les packages que l'on va utiliser et les options générales pour l'édition du document.

La publication des documents peut se faire en différents formats : html, word, pdf (via latex), présentation html (via revealjs), présentation powerpoint, présentation beamer(via latex), ... On se référera au site de [Quarto](#) pour plus de détails.

Ici, on commence simplement avec quelques manipulations pour comprendre l'environnement de travail, puis on verra comment charger des données sous différents formats.

## 1.2 Premières manipulations

- Créer un document script (.R) : pour simplement éditer du code
- Créer un document quarto (.qmd) : pour mixer du code et du texte
- Commenter du code
- Afficher de l'aide sur une fonction

## 1.3 Charger des données

### 1.3.1 Un tableau de données

Fichier .csv, .xlsx, .rds

```
data<-read.csv("le/chemin/de/mon/fichier.csv")

library(readxl)
data<-read_xlsx("le/chemin/de/mon/fichier.xlsx")
```

```
library(readr)
data<- read_rds("le/chemin/de/mon/fichier.rds")
```

### 1.3.2 Une collection de fichier textes

Un dossier avec plusieurs fichier .txt ou .docx ou .pdf

```
library(readtext)
#Exemple de nom de document : "int1_2024_dirigeant.txt"
data<-readtext("le/chemin/de/mes/fichiers/*.txt",
               docvarsfrom = "filenames",
               docvarnames = c("int", "année", "type"),
               dvsep = "_")

data<-readtext("le/chemin/de/mes/fichiers/*.docx")

data<-readtext("le/chemin/de/mes/fichiers/*.pdf")
```

Pour les problèmes de mise en forme, on consultera la vignette du [package readtext](#).

Une autre solution pour les fichiers pdf, permettant d'enlever tous les éléments de mise en forme :

```
library(tm)
#on récupère les noms des fichiers à lire depuis les dossiers
files <- list.files( pattern = "pdf$", recursive = T, include.dirs = T)

#on lit les fichiers, sans la mise en forme
corp<-Corpus(URISource(files),
             readerControl = list(reader = readPDF, text=("-layout")))
#on enlève les sauts de page et autres mises en forme à partir d'espace
corp <- tm_map(corp, stripWhitespace)
```

Un autre outil pour les pdf : le package [pdftools](#).

### 1.3.3 Reconnaissance Optique des Caractères (OCR)

Pour ça, on utilise le package [tesseract](#) :



En fondant la *Revue Internationale de l'Etalage*, nous avons cédé à une ambition qui peut s'exprimer en trois mots : *Faire œuvre utile*.

Faire œuvre utile en facilitant la tâche du commerçant détaillant par une documentation sérieuse et raisonnée se rapportant aux choses de sa profession, et notamment aux meilleurs procédés pour attirer la clientèle.

Faire œuvre utile surtout en vulgarisant pratiquement l'art appliqué à l'Etalage.

Si notre carrière de publiciste s'honore sans fausse modestie d'avoir doté d'autres corporations de revues professionnelles qui rendent des services, nous devons déclarer en toute sincérité que la fondation d'aucune de celles-ci ne répondait à un tel besoin.

Aussi est-ce d'un pied ferme que nous nous engageons dans le vaste champ d'actions qui s'ouvre devant nous.

*L'Etalage!*... Quel sujet intéressant et captivant au premier chef. N'est-il pas vraiment incompréhensible qu'aucune plume autorisée n'ait jamais tenté, en France, d'en établir les règles, d'en vulgariser les principes. Et pourtant, en envisageant la chose à un point de vue général, ne peut-on pas dire que l'Etalage est un des ornements de nos cités qui met sans cesse en évidence, aux yeux de tous, la supériorité du goût et les aptitudes artistiques d'un pays ! Tandis qu'au point de vue de leurs propres intérêts, nos lecteurs ne doivent-ils pas considérer l'Etalage comme *l'aimant qui attire les affaires* !

On le comprend si bien aujourd'hui que le commerçant qui se désintéresse de son étalage est une exception appelée de plus en plus à disparaître.

Et cependant, parmi ceux qui se rendent un compte exact de l'importance du rôle de l'Etalage dans les affaires de détail, combien peu possèdent à fond l'art de présenter leurs produits sous leur meilleur jour et de la façon la plus favorable.

Quoi de surprenant d'ailleurs à cela. En

Exemple avec cette image :

```
library(tesseract)
tesseract_download("fra") #pour télécharger le modèle de langage

text <- tesseract::ocr("N1_avril1909b.jpeg", engine = "fra")

cat(text) #pour afficher le texte avec sa mise en page
```

## 2 Pré-traitements

### 2.1 Introduction

Le document de travail contient deux types d'éléments : du texte pour expliquer et présenter ce que l'on fait et du code pour réaliser les manipulations de données, les analyses et les graphiques. On commence toujours un document avec un bloc de code de setup, pour lister et charger les packages que l'on va utiliser et les options générales pour l'édition du document.

```
library(readxl)
library(tidyverse)
library(quanteda)
library(quanteda.textstats)
library(quanteda.textplots)
library(RColorBrewer)
```

Dans un premier temps, nous allons tout simplement charger la base de données de travail puis la décrire. Ensuite, nous créerons un corpus, le visualisons. Puis nous effectueront quelques analyses liminaires, avant de voir les pré-traitements à réaliser sur le corpus.

### 2.2 Les données

```
#On charge les données, stockées dans un fichier csv

data <- read_csv("data/data_trustpilot_oiseaux.csv")
```

```
Rows: 4388 Columns: 7
-- Column specification -----
Delimiter: ","
chr (4): auteur, date, month, comments
dbl (3): id, year, note

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
names(data)
```

```
[1] "id"      "auteur"  "date"    "month"   "year"    "note"    "comments"
```

```
view(data)
```

```
data
```

```
# A tibble: 4,388 x 7
```

	id	auteur	date	month	year	note	comments
	<dbl>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<chr>
1	1	MAQUET Cyril	07 août 2023	août	2023	5	"Comme to~
2	2	Mme Laurence Wolff	08 août 2023	août	2023	5	"Le délai~
3	3	Une nouvelle cliente	07 août 2023	août	2023	5	"Produits~
4	4	Patricia ALLAMAN	15 août 2023	août	2023	5	"Envoi ra~
5	5	VPL	24 juillet 2023	juillet	2023	5	"Expéditi~
6	6	PHILIPPE GODIN	08 août 2023	août	2023	5	"site sér~
7	7	Mme MARIA ADILIA PEREIRA	15 août 2023	août	2023	1	"deux sem~
8	8	Rachel Mattyssen	31 juillet 2023	juillet	2023	5	"Très bie~
9	9	Estelle Fay	16 juillet 2023	juillet	2023	5	"Enfin un~
10	10	Mme T.	06 août 2023	août	2023	4	"Satisfai~

```
# i 4,378 more rows
```

```
#Résumé des données
```

```
summary(data)
```

	id	auteur	date	month
Min.	: 1	Length:4388	Length:4388	Length:4388
1st Qu.	:1098	Class :character	Class :character	Class :character
Median	:2194	Mode :character	Mode :character	Mode :character
Mean	:2194			
3rd Qu.	:3291			
Max.	:4388			

	year	note	comments
Min.	:2013	Min. :1.000	Length:4388
1st Qu.	:2018	1st Qu.:5.000	Class :character
Median	:2020	Median :5.000	Mode :character
Mean	:2019	Mean :4.641	
3rd Qu.	:2021	3rd Qu.:5.000	
Max.	:2023	Max. :5.000	

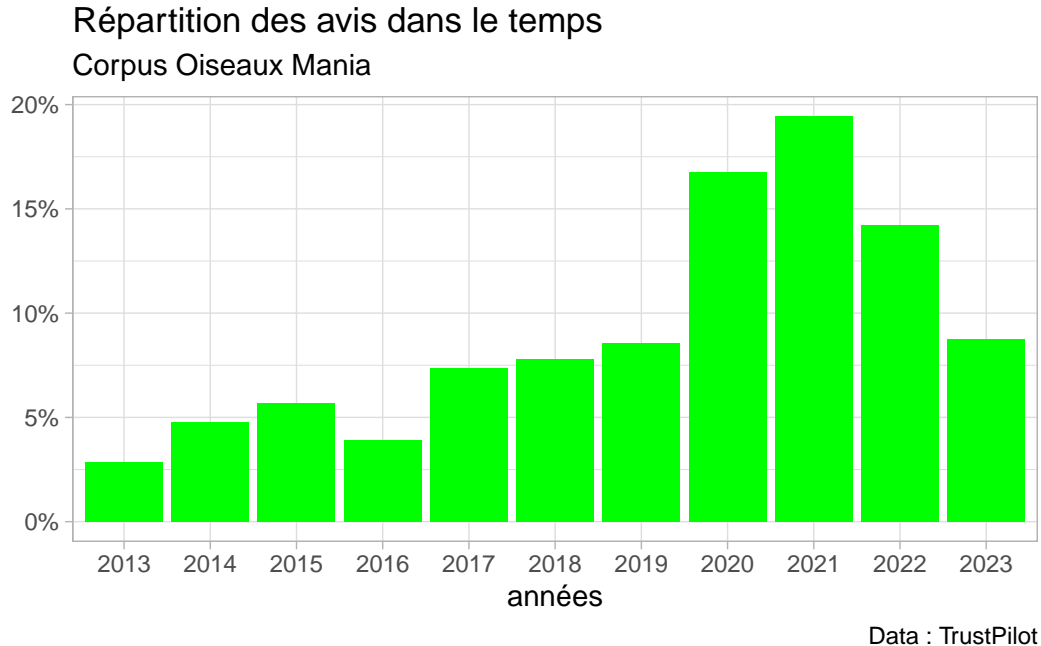
## 2.3 Premières analyses des données

Avant de s'intéresser au contenu des commentaires, explorons la structure des données. On va regarder la distribution des commentaires et des notes dans le temps, et s'intéresser à la longueur des avis clients.

```
#Les années
data$year<-as.factor(data$year)
summary(data$year)
```

```
2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023
  125   209   249   171   324   341   375   735   853   623   383
```

```
data%>%
  group_by(year)%>%
  summarise(n=n(), prop=n/nrow(data))%>%
  ggplot(aes(year,prop))+
  geom_col(fill="green",show.legend = TRUE)+
  scale_y_continuous(labels = scales::percent)+
  theme_light()+
  labs(title = "Répartition des avis dans le temps", subtitle = "Corpus Oiseaux Mania", capt
```



```
#Les notes
summary(data$note)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1.000   5.000   5.000   4.641   5.000   5.000
```

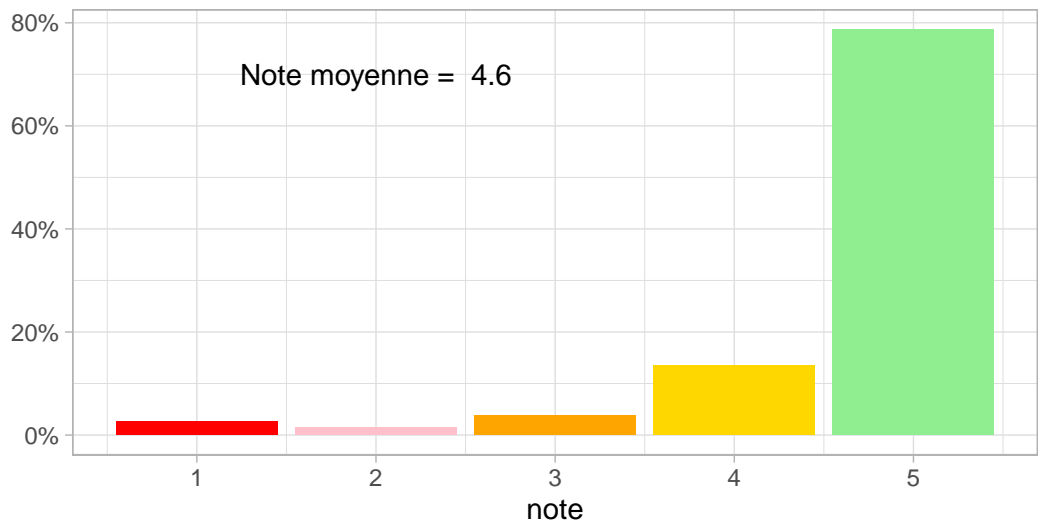
```
summary(as.factor(data$note))
```

```
 1    2    3    4    5
116   62  169  589 3452
```

```
data%>%
  group_by(note)%>%
  summarise(n=n(), prop=n/nrow(data))%>%
  ggplot(aes(note,prop))+
  geom_col(fill=c("red","pink","orange","gold","lightgreen"))+
  annotate("text", x=2, y=0.7, label=paste("Note moyenne = ",round(mean(data$note),1)))+
  scale_y_continuous(labels=scales::percent)+
  theme_light()+
  labs(title = "Répartition des avis en fonction des notes", subtitle = "Corpus Oiseaux Mania")
```

## Répartition des avis en fonction des notes

### Corpus Oiseaux Mania

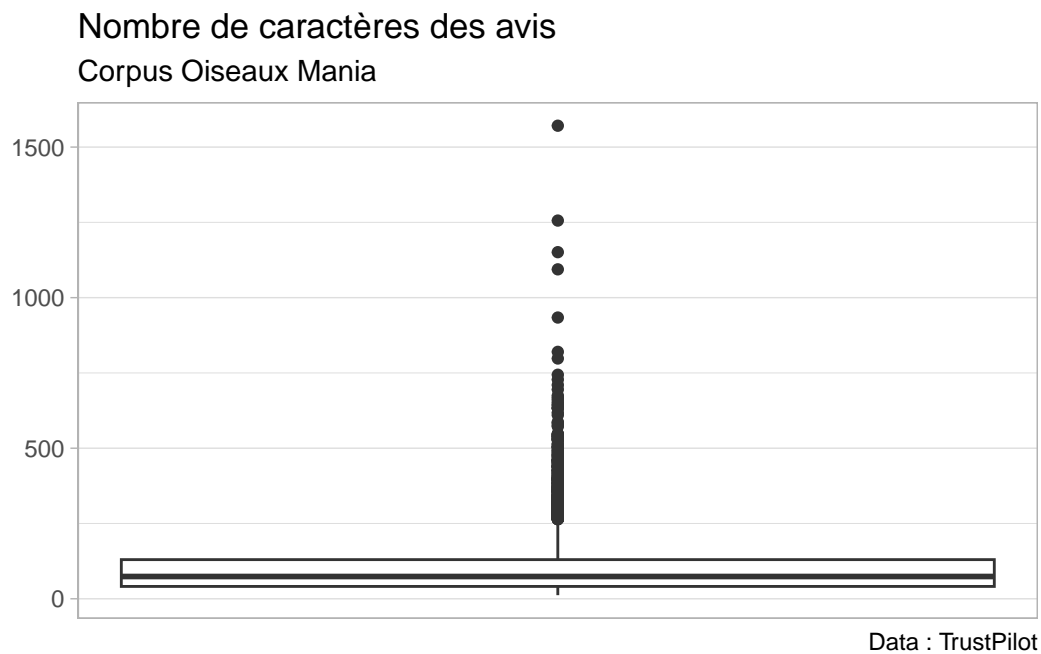


Data : TrustPilot

```
#Le nombre de caractère
data$nb_caractere<-nchar(data$comments) #on compte le nombre de caractère de chaque commenta.
summary(data$nb_caractere)
```

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
12.0   41.0   74.0  104.3  130.0  1571.0
```

```
ggplot(data, aes(nb_caractere))+
  geom_boxplot()+
  scale_y_continuous(NULL, breaks = NULL)+
  labs(x=NULL,title = "Nombre de caractères des avis", subtitle = "Corpus Oiseaux Mania", cap
  theme_light()+
  coord_flip()
```



```
#Le nombre de tokens
data$nb_token<-ntoken(data$comments) #on compte le nombre de caractère de chaque commentaire
```

Warning: ntoken.character()/ntype.corpus() was deprecated in quanteda 4.0.0.  
i Please use ntoken(tokens(x)) instead.

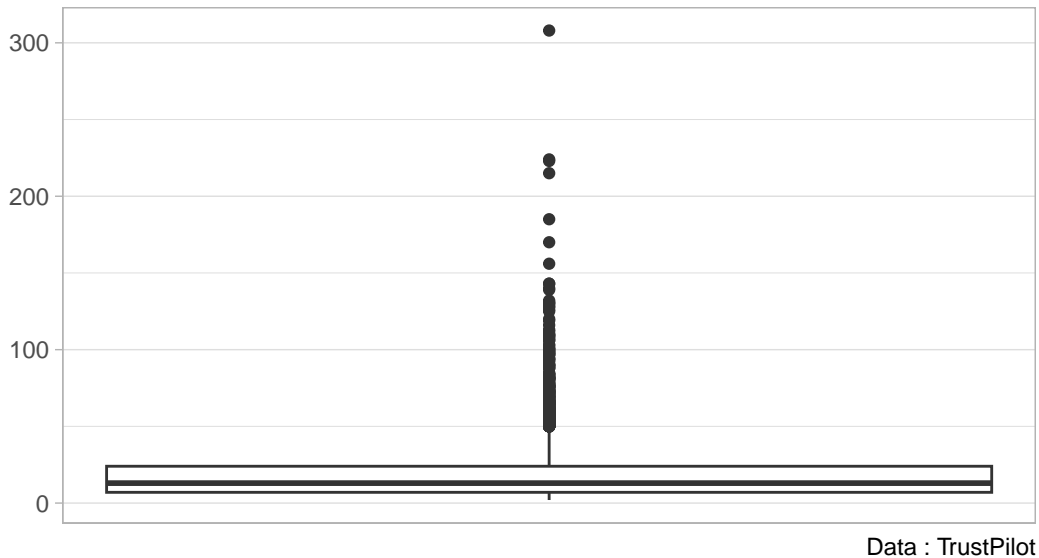
```
summary(data$nb_token)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
2.0	7.0	13.0	19.3	24.0	308.0

```
ggplot(data, aes(nb_token))+  
  geom_boxplot()+  
  scale_y_continuous(NULL, breaks = NULL)+  
  labs(x=NULL, title = "Nombre de tokens des avis", subtitle = "Corpus Oiseaux Mania", caption = "Data : TrustPilot") +  
  theme_light()+  
  coord_flip()
```

## Nombre de tokens des avis

Corpus Oiseaux Mania

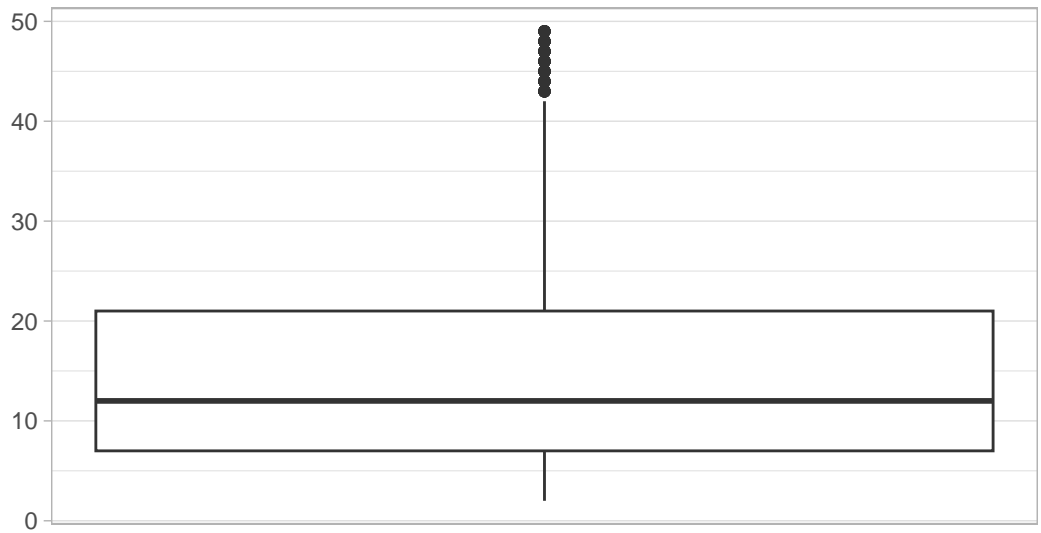


```
#On va filtrer au-dessus de 100 tokens  
data_100t<-data%>%filter(nb_token<50)  
  
ggplot(data_100t, aes(nb_token))+  
  geom_boxplot()+  
  scale_y_continuous(NULL, breaks = NULL)+  
  labs(x=NULL, title = "Nombre de tokens des avis", subtitle = "Corpus Oiseaux Mania", caption = "Data : TrustPilot") +  
  theme_light()+  
  coord_flip()
```



## Nombre de tokens des avis

Corpus Oiseaux Mania



Data : TrustPilot

```
#Les notes dans le temps
```

```
data%>%
```

```
  mutate(note=as.factor(note))%>%
```

```
  group_by(year, note)%>%
```

```
  summarise(n=n() ,prop=n/nrow(data))%>%
```

```
  ggplot(aes(year, prop))+
```

```
  geom_col(aes(fill=note), show.legend = FALSE)+
```

```
  scale_fill_discrete(type=c("red","pink","orange","gold","lightgreen"))+
```

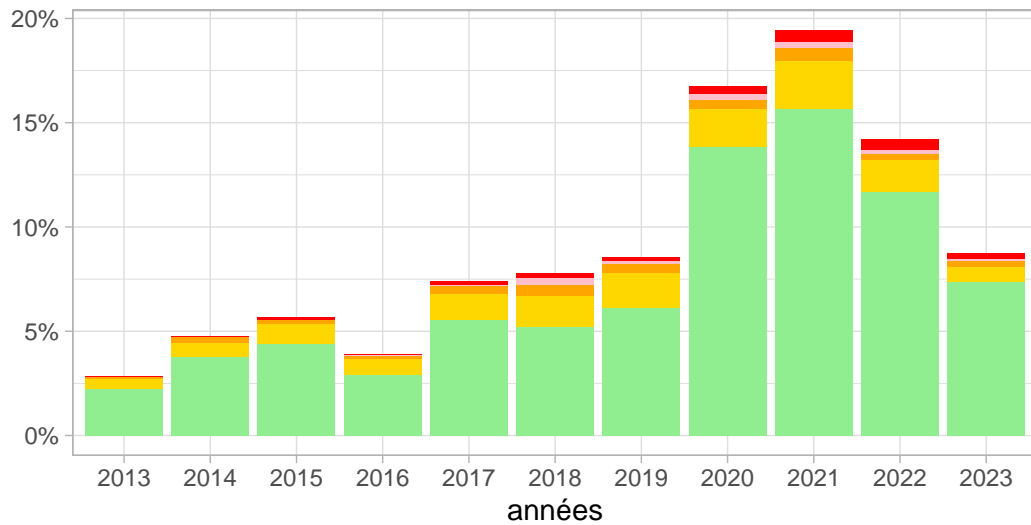
```
  scale_y_continuous(labels=scales::percent)+
```

```
  theme_light()+
```

```
  labs(title = "Répartition des avis dans le temps", subtitle = "Corpus Oiseaux Mania", capt
```

## Répartition des avis dans le temps

Corpus Oiseaux Mania

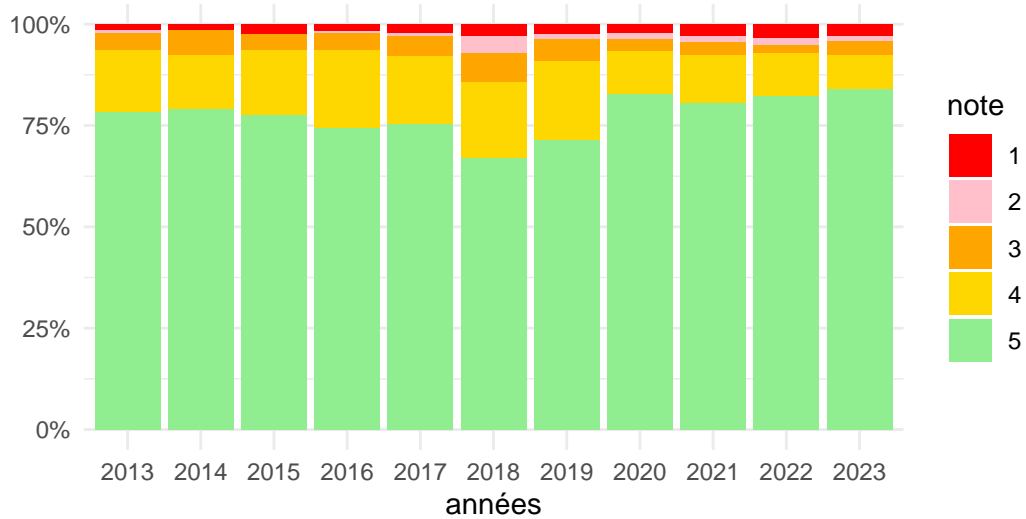


Data : TrustPilot

```
data%>%
  mutate(note=as.factor(note))%>%
  group_by(year, note)%>%
  summarise(n=n())%>%
  ggplot(aes(x=year, y=n, group=note))+
  geom_bar(position="fill",stat="identity", aes(fill=note))+
  scale_fill_discrete(type=c("red","pink","orange","gold","lightgreen"))+
  scale_y_continuous(labels=scales::percent)+
  theme_minimal()+
  labs(title = "Comparaison de la répartition des notes dans le temps", subtitle = "Corpus Oiseaux Mania")
```

## Comparaison de la répartition des notes dans le temps

### Corpus Oiseaux Mania



Data : TrustPilot

### 3 Création du corpus et premières observations du corpus

Tout d'abord, nous transformons le jeu de données en corpus. La variable qui contient le texte est "comments", les autres variables vont devenir des métadonnées du corpus, c'est-à-dire des variables associées à chaque texte. Cela sera utile par la suite pour faire des analyses comparatives entre les textes suivant différentes variables (le temps en particulier, mais pas seulement).

```
#Création du corpus
corpus_oiseaux<-corpus(data, text_field = "comments")
corpus_oiseaux
```

Corpus consisting of 4,388 documents and 8 docvars.

text1 :

"Comme toujours, super service. Ne changez rien!(sauf peut êt..."

text2 :

"Le délai de ma commande super rapide Le délais des ma comman..."

text3 :

"Produits de qualité et équipe professionnelle ... Très bon acc..."

text4 :

"Envoi rapide, bien emballé et conforme à l'annonce"

text5 :

"Expédition internationale ! J'ai récemment déménagé en Espag..."

text6 :

"site sérieux bon produit livraisons plus que correct mais pe..."

[ reached max\_ndoc ... 4,382 more documents ]

```
corpus_oiseaux["text600"] #pour visualiser un texte précis
```

Corpus consisting of 1 document and 8 docvars.

text600 :

"Commande reçu dans les temps pas de surprise. Très bon site"

```
a<-corpus_oiseaux["text30"]  
rm(a)
```

Ensuite, nous allons extraire de chaque texte les termes qui les composent. Ces termes sont nommés “token” (jeton), et comme vous pouvez le voir, ce ne sont pas uniquement des mots, mais tout caractère ou suite de caractères séparés des autres par un espace.

```
#Extraction des tokens  
tok<-tokens(corpus_oiseaux)  
tok["text600"]
```

Tokens consisting of 1 document and 8 docvars.

text600 :

[1]	"Commande"	"reçu"	"dans"	"les"	"temps"	"pas"
[7]	"de"	"surprise"	".."	"Très"	"bon"	"site"

Chaque texte est maintenant décomposé en une suite de tokens. Pour voir les termes les plus fréquents dans le corpus, ainsi que leur co-occurrences (apparition de deux termes en même temps), il convient de transformer l’objet tok en une matrice termes-documents. En ligne, tous les tokens identifiés, en ligne, tous les textes du corpus, et les valeurs correspondent au nombre d’occurrences (d’apparitions) de chaque token dans chaque document. Une particularité de cette matrice est qu’elle contient énormément de zéro.

```
#Transformation en document-term frequency matrix  
dfm<-dfm(tok)  
dfm
```

Document-feature matrix of: 4,388 documents, 5,658 features (99.72% sparse) and 8 docvars.

```
features  
docs      comme toujours , super service . ne changez rien !  
text1      1          1 1      1          1 4 1          1      1 1  
text2      0          1 1      2          0 3 0          0      0 0  
text3      0          0 0      0          0 2 0          0      0 0  
text4      0          0 1      0          0 0 0          0      0 0
```

```

text5      0      0 4      0      0 2 1      0      0 4
text6      0      0 0      0      0 0 0      0      0 0
[ reached max_ndoc ... 4,382 more documents, reached max_nfeat ... 5,648 more features ]

```

Enfin, nous pouvons avoir un aperçu des termes les plus fréquents. Nous les visualisons d'abord sous forme de tableau (les 20 tokens les plus fréquents), puis sous la forme d'un nuage de mots, où la taille des mots correspond à leur fréquence dans le corpus.

```

#Visualisation des termes les plus fréquents
textstat_frequency(dfm, n=20) #les 20 premiers termes les plus fréquents

```

	feature	frequency	rank	docfreq	group
1	.	4777	1	2094	all
2	,	2788	2	1576	all
3	de	2685	3	1646	all
4	et	2530	4	1915	all
5	très	1996	5	1529	all
6	rapide	1659	6	1569	all
7	je	1622	7	1221	all
8	livraison	1412	8	1299	all
9	!	1349	9	649	all
10	à	1312	10	1022	all
11	commande	1238	11	1030	all
12	le	1215	12	812	all
13	la	1215	12	868	all
14	pour	1049	14	782	all
15	les	1013	15	754	all
16	bien	920	16	803	all
17	en	801	17	618	all
18	site	793	18	686	all
19	produits	738	19	661	all
20	un	729	20	565	all

```

textplot_wordcloud(dfm) #nuage de mots

```

```

Warning in text.default(x1, y1, word[i], cex = (1 + adjust) * size[i], offset =
0, : font metrics unknown for Unicode character U+1F44D

```



Pour conclure sur cette première approche du corpus, nous voyons que nos analyses sont gênées par la présence de la ponctuation et de plein de petits mots “vides de sens” (les articles par exemple). C’est pourquoi nous allons nettoyer le corpus pour avoir une meilleure vision de ce qu’il contient.

## 4 Nettoyage du corpus

Le nettoyage du corpus pour les analyses se fait lors de la transformation en tokens. Nous allons ajouter des options pour supprimer la ponctuation, les chiffres et les stopwords (les mots qui n'apportent pas de sens sémantique mais permettent l'articulation du discours).

```
stopwords("fr")
```

[1]	"au"	"aux"	"avec"	"ce"	"ces"	"dans"
[7]	"de"	"des"	"du"	"elle"	"en"	"et"
[13]	"eux"	"il"	"je"	"la"	"le"	"leur"
[19]	"lui"	"ma"	"mais"	"me"	"même"	"mes"
[25]	"moi"	"mon"	"ne"	"nos"	"notre"	"nous"
[31]	"on"	"ou"	"par"	"pas"	"pour"	"qu"
[37]	"que"	"qui"	"sa"	"se"	"ses"	"son"
[43]	"sur"	"ta"	"te"	"tes"	"toi"	"ton"
[49]	"tu"	"un"	"une"	"vos"	"votre"	"vous"
[55]	"c"	"d"	"j"	"l"	"à"	"m"
[61]	"n"	"s"	"t"	"y"	"été"	"étée"
[67]	"étéés"	"étés"	"étant"	"suis"	"es"	"est"
[73]	"sommes"	"êtes"	"sont"	"serai"	"seras"	"sera"
[79]	"serons"	"serez"	"seront"	"serais"	"serait"	"serions"
[85]	"seriez"	"seraient"	"étais"	"était"	"étions"	"étiez"
[91]	"étaient"	"fus"	"fut"	"fûmes"	"fûtes"	"furent"
[97]	"sois"	"soit"	"soyons"	"soyez"	"soient"	"fusse"
[103]	"fusses"	"fût"	"fussions"	"fussiez"	"fussent"	"ayant"
[109]	"eu"	"eue"	"eues"	"eus"	"ai"	"as"
[115]	"avons"	"avez"	"ont"	"aurai"	"auras"	"aura"
[121]	"aurons"	"aurez"	"auront"	"aurais"	"aurait"	"aurions"
[127]	"auriez"	"auraient"	"avais"	"avait"	"avons"	"aviez"
[133]	"avaient"	"eut"	"eûmes"	"eûtes"	"eurent"	"aie"
[139]	"aies"	"ait"	"ayons"	"ayez"	"aient"	"eusse"
[145]	"eusses"	"eût"	"eussions"	"eussiez"	"eussent"	"ceci"
[151]	"cela"	"celà"	"cet"	"cette"	"ici"	"ils"
[157]	"les"	"leurs"	"quel"	"quels"	"quelle"	"quelles"
[163]	"sans"	"soi"				



```
tok<-tokens(corpus_oiseaux, remove_punct = TRUE, remove_numbers = TRUE)%>%
  tokens_remove(stopwords("fr"))
corpus_oiseaux["text600"]
```

Corpus consisting of 1 document and 8 docvars.

text600 :

"Commande reçu dans les temps pas de surprise. Très bon site"

```
tok["text600"]
```

Tokens consisting of 1 document and 8 docvars.

text600 :

[1] "Commande" "reçu" "temps" "surprise" "Très" "bon" "site"

Ensuite, on transforme en dfm et on visualise ce que ça donne.

```
dfm<-dfm(tok)
dfm
```

Document-feature matrix of: 4,388 documents, 5,430 features (99.81% sparse) and 8 docvars.

```
features
docs  comme toujours super service changez rien sauf peut être là
text1  1          1      1          1          1  1  1  1  1  1
text2  0          1      2          0          0  0  0  0  0  1
text3  0          0      0          0          0  0  0  0  0  0
text4  0          0      0          0          0  0  0  0  0  0
text5  0          0      0          0          0  0  0  0  0  0
text6  0          0      0          0          0  0  0  0  0  0
```

[ reached max\_ndoc ... 4,382 more documents, reached max\_nfeat ... 5,420 more features ]

```
textstat_frequency(dfm, n=20)
```

	feature	frequency	rank	docfreq	group
1	très	1996	1	1529	all
2	rapide	1659	2	1569	all
3	livraison	1412	3	1299	all
4	commande	1238	4	1030	all
5	bien	920	5	803	all
6	site	793	6	686	all

7	produits	738	7	661	all
8	a	706	8	566	all
9	produit	676	9	609	all
10	bon	639	10	575	all
11	merci	553	11	529	all
12	recommande	553	11	534	all
13	parfait	522	13	474	all
14	qualité	487	14	447	all
15	j'ai	382	15	289	all
16	oiseaux	379	16	319	all
17	tout	375	17	347	all
18	reçu	362	18	332	all
19	colis	360	19	313	all
20	rapidement	356	20	343	all

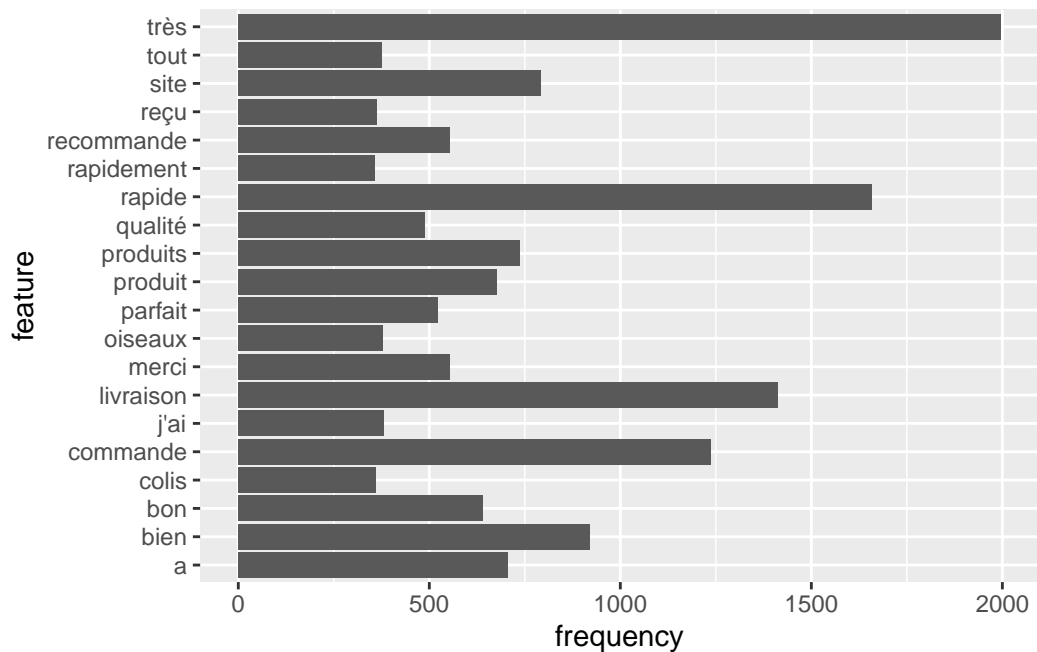
```
textplot_wordcloud(dfm)
```

Warning in text.default(x1, y1, word[i], cex = (1 + adjust) \* size[i], offset = 0, : font metrics unknown for Unicode character U+1F44D

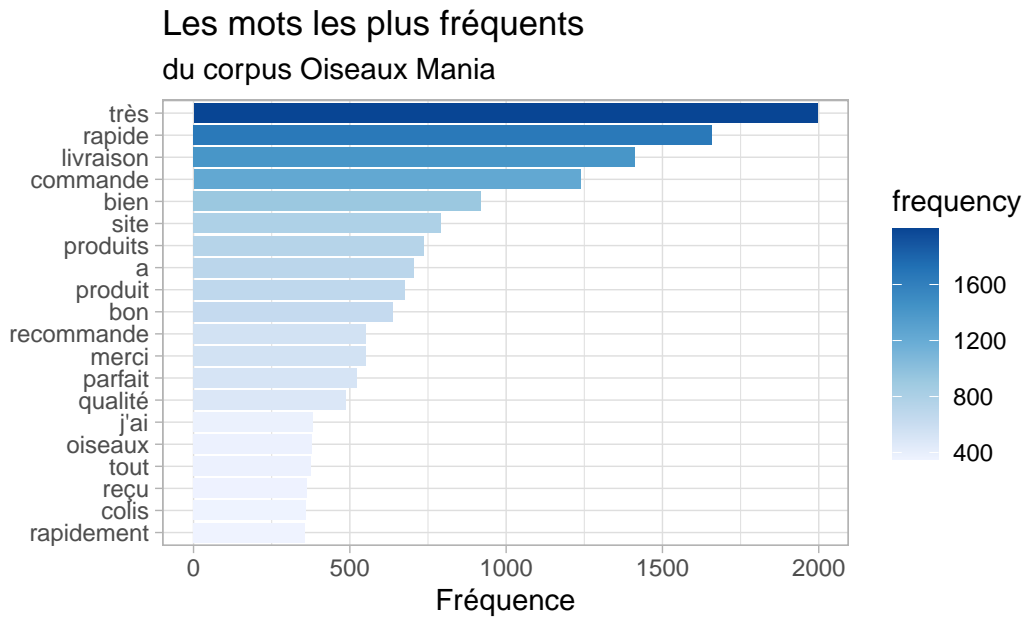


```
g<-textstat_frequency(dfm,n=20)
```

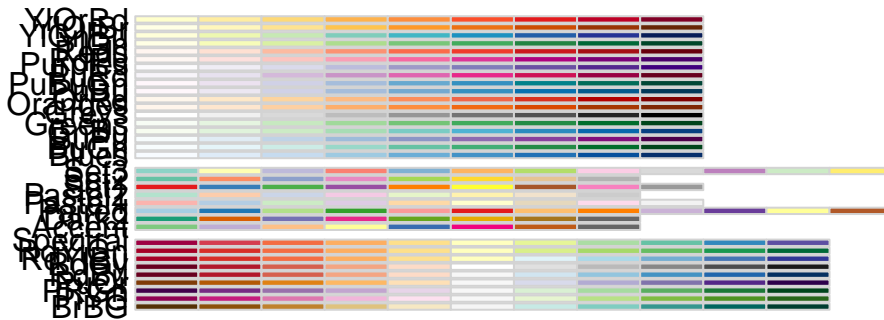
```
ggplot(g, aes(x = feature, y=frequency))+  
  geom_col()+  
  coord_flip()
```



```
ggplot(g, aes(x = reorder(feature, frequency), y=frequency, fill=frequency))+  
  geom_col(show.legend = TRUE)+  
  coord_flip()+  
  theme_light()+  
  scale_fill_distiller(palette = "Blues", direction = 1)+  
  labs(title="Les mots les plus fréquents", subtitle = "du corpus Oiseaux Mania", caption = "S"  
  xlab(NULL)+  
  ylab("Fréquence")
```



```
display.brewer.all()
```



Globalement, la commande et la livraison sont TRÈS rapides et les produits sont bons. La surreprésentation de ces termes dans le corpus nous empêche de voir les thématiques abordées

de manière moins évidentes. Nous avons plusieurs solutions qui s’offrent à nous : filtrer les mots trop fréquents du corpus ou nous intéresser à une autre mesure de la fréquence d’apparition. Nous allons d’abord filtrer le corpus.

On peut aussi vouloir remplacer des termes par d’autres, comme ici “produits” par “produit”.

```
tok<-tokens_replace(tok, "produits", "produit")
dfm<-dfm(tok)
textstat_frequency(dfm, n=20)
```

	feature	frequency	rank	docfreq	group
1	très	1996	1	1529	all
2	rapide	1659	2	1569	all
3	produit	1414	3	1243	all
4	livraison	1412	4	1299	all
5	commande	1238	5	1030	all
6	bien	920	6	803	all
7	site	793	7	686	all
8	a	706	8	566	all
9	bon	639	9	575	all
10	merci	553	10	529	all
11	recommande	553	10	534	all
12	parfait	522	12	474	all
13	qualité	487	13	447	all
14	j'ai	382	14	289	all
15	oiseaux	379	15	319	all
16	tout	375	16	347	all
17	reçu	362	17	332	all
18	colis	360	18	313	all
19	rapidement	356	19	343	all
20	service	351	20	325	all

## 5 Filtrer le corpus des termes trop fréquents

Nous allons filtrer les mots qui sont présents plus de 500 fois dans le corpus.

```
dfm_trim<-dfm_trim(dfm, max_termfreq = 500)

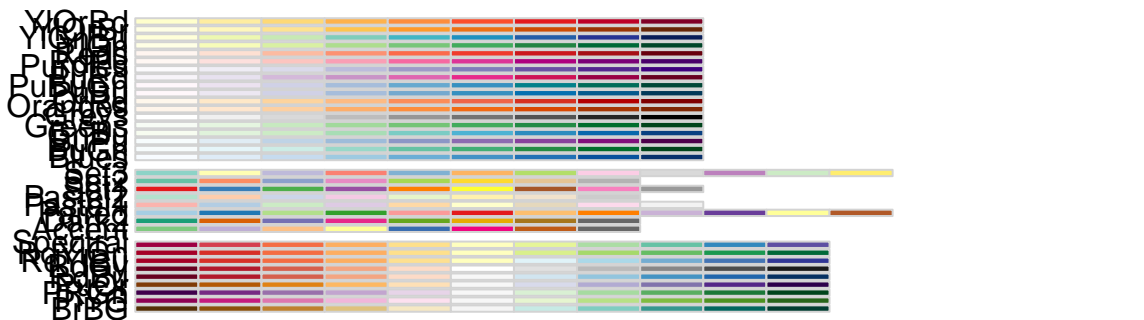
textstat_frequency(dfm_trim, n=20)
```

	feature	frequency	rank	docfreq	group
1	qualité	487	1	447	all
2	j'ai	382	2	289	all
3	oiseaux	379	3	319	all
4	tout	375	4	347	all
5	reçu	362	5	332	all
6	colis	360	6	313	all
7	rapidement	356	7	343	all
8	service	351	8	325	all
9	prix	327	9	307	all
10	satisfaite	327	9	301	all
11	super	326	11	294	all
12	conforme	326	11	307	all
13	rien	317	13	299	all
14	bonne	293	14	277	all
15	plus	292	15	254	all
16	tres	278	16	216	all
17	toujours	274	17	236	all
18	mania	256	18	228	all
19	envoi	239	19	231	all
20	sérieux	232	20	220	all

```
textplot_wordcloud(dfm_trim, max_words = 100, color = rev(brewer.pal(10, "Set2")))
```

Warning in brewer.pal(10, "Set2"): n too large, allowed maximum for palette Set2 is 8  
Returning the palette you asked for with that many colors

```
display.brewer.all()
```



Une autre manière de s’y prendre est d’éliminer directement les termes que l’on ne veut pas voir apparaître.

```
textstat_frequency(dfm,n=20)
```

	feature	frequency	rank	docfreq	group
1	très	1996	1	1529	all
2	rapide	1659	2	1569	all
3	produit	1414	3	1243	all
4	livraison	1412	4	1299	all
5	commande	1238	5	1030	all
6	bien	920	6	803	all
7	site	793	7	686	all
8	a	706	8	566	all
9	bon	639	9	575	all
10	merci	553	10	529	all
11	recommande	553	10	534	all
12	parfait	522	12	474	all
13	qualité	487	13	447	all
14	j'ai	382	14	289	all
15	oiseaux	379	15	319	all
16	tout	375	16	347	all
17	reçu	362	17	332	all
18	colis	360	18	313	all
19	rapidement	356	19	343	all
20	service	351	20	325	all

```
rem<-c("très","rapide","produit","livraison", "commande", "bien", "site", "a", "bon", "merci"
```

```
dfm_rem<-dfm_remove(dfm, rem)
textstat_frequency(dfm_rem, n=20)
```

	feature	frequency	rank	docfreq	group
1	qualité	487	1	447	all
2	oiseaux	379	2	319	all
3	tout	375	3	347	all
4	reçu	362	4	332	all
5	colis	360	5	313	all
6	rapidement	356	6	343	all
7	service	351	7	325	all
8	prix	327	8	307	all
9	satisfaite	327	8	301	all
10	super	326	10	294	all
11	conforme	326	10	307	all

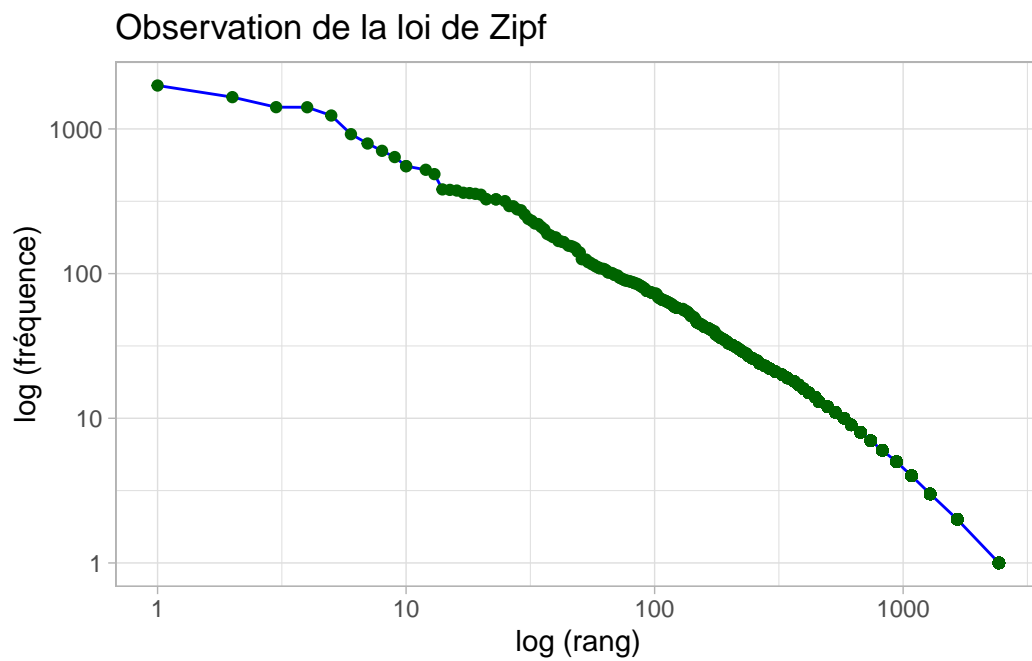


12	rien	317	12	299	all
13	bonne	293	13	277	all
14	plus	292	14	254	all
15	toujours	274	15	236	all
16	mania	256	16	228	all
17	envoi	239	17	231	all
18	sérieux	232	18	220	all
19	choix	222	19	205	all
20	satisfait	220	20	205	all

## 6 La loi de Zipf

Vérifions la proposition de la loi de Zipf, selon laquelle la fréquence d'apparition d'un terme est inversement proportionnel à son rang.

```
zipf<-textstat_frequency(dfm)
ggplot(zipf, aes(rank, frequency))+
  geom_line(color="blue")+
  geom_point(color="darkgreen")+
  scale_x_log10() +
  scale_y_log10()+
  theme_light()+
  labs(title = "Observation de la loi de Zipf",x="log (rang)",y="log (fréquence)")
```



## 7 Mesures de fréquence

On s'est pour l'instant intéressé uniquement aux termes les plus fréquents dans un corpus. On a vu comment éliminer les termes trop fréquents pour qu'ils nous apportent de l'information. Pour l'analyse de topics, il nous faut prendre un autre angle d'attaque : afin de détecter les sujets abordés dans un corpus, on ne peut se contenter d'observer les mots les plus fréquents, il faut s'intéresser aux termes dont la fréquence dans l'ensemble du corpus est faible, mais qui contribuent fortement à différencier les éléments du corpus entre eux (les documents). On utilise pour cela une mesure de fréquence pondérée : la *tf-idf* pour *term frequency - inverse document frequency* qui permet d'accorder plus de poids aux termes les plus discriminants du corpus.  $tf-idf = \frac{\text{occurrence du mot dans le document}}{\text{nombre de mots dans le document}} * \log\left(\frac{\text{nombre de documents dans le corpus}}{\text{nombre de documents dans lequel le mot apparaît}}\right)$

### 7.1 Pondération tf-idf

On commence par reprendre nos manipulations précédentes : création de corpus, élimination des stopwords, constitution de bi- ou tri- grammes. On applique ensuite la pondération tf-idf.

```
dfmtfidf<-dfm_tfidf(dfm)
```

```
dfmtfidf
```

Document-feature matrix of: 4,388 documents, 5,429 features (99.81% sparse) and 8 docvars.  
features

docs	comme	toujours	super	service	changez	rien	sauf	peut
text1	1.406738	1.269355	1.173919	1.130383	3.040207	1.166595	2.320047	1.926263
text2	0	1.269355	2.347839	0	0	0	0	0
text3	0	0	0	0	0	0	0	0
text4	0	0	0	0	0	0	0	0
text5	0	0	0	0	0	0	0	0
text6	0	0	0	0	0	0	0	0

features

docs	être	là
text1	1.961025	2.320047
text2	0	2.320047
text3	0	0





```
#On filtre les mots trop fréquents
```

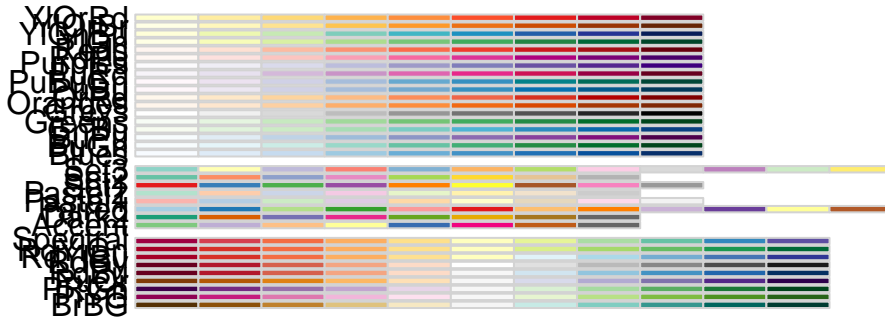
```
dfm_trim<-dfm_trim(dfm, max_termfreq = 500)
```

```
dfmtfidf_trim<-dfm_tfidf(dfm_trim)
```

```
textplot_wordcloud(dfm_trim, max_words = 200)
```



```
display.brewer.all()
```



## 8 Comprendre le sens des termes

On peut visualiser un ou plusieurs termes dans leur contexte, afin d’avoir une meilleure compréhension de leur sens. Pour cela on utilise la fonction “kwic” pour key word in context, à partir de l’objet tokens :

```
head(kwic(tok,"livraison",window = 3))
```

Keyword-in-context with 6 matches.

[text2, 12]	revanche n'arrive cliquer	livraison	
[text3, 13]	depuis conseil jusqu'à	livraison	
[text5, 36]	perfection puisque délai	livraison	
[text8, 14]	a entière satisfaction	livraison	
[text14, 1]		Livraison	
[text15, 21]	trouve peut frais	livraison	

adresse toujours là  
produit recommande vivement  
Espagne depuis France  
aussi a rapide  
rapide soignée Bons  
élevé

```
# kwic(tok,"livraison",window = 3)
```

```
head(kwic(tok, c("commande", "recommande"),window = 3))
```

Keyword-in-context with 6 matches.

[text2, 2]	délai	commande	super rapide délais
[text2, 6]	super rapide délais	commande	super rapide revanche
[text3, 15]	jusqu'à livraison produit	recommande	vivement oiseaux Mania
[text5, 14]	Food regardant dernière	commande	j'avais faite France
[text7, 5]	semaines j'ai passé	commande	toujours livré étoile
[text8, 9]	j'ai néanmoins fait	commande	difficulté a entière



```
head(kwic(tok,"perroquet",window = 3))
```

Keyword-in-context with 6 matches.

```
[text5, 51]    d'avoir trouvé solution | perroquet |
[text18, 8]     compétent sers site | perroquet |
[text22, 18]    convient tout fait | perroquet |
[text34, 6]     treats granulés compressés | Perroquet |
[text37, 10]    choix produit petit | perroquet |
[text39, 2]     Alimentation | perroquet |
```

```
afin qu'il puisse
  Excellent rapide
très heureux aussi
Gris Gabon très
n'hésiterai refaire commandes
commande depuis quelques
```

```
head(kwic(tok,"prix",window=10))
```

Keyword-in-context with 6 matches.

```
[text14, 6]
[text15, 15]
[text26, 23]
[text28, 12]
[text31, 77]
[text32, 7]
```

```

                                Livraison rapide soignée Bons produit
      rapide satisfait commander cher oiseaux mania car Besançon trouve produit
beaucoup choix graines friandises accessoires commande a livrée rapidement encombres
      harnais reçus déplore juste l'emballage spartiate arrivé moitié déchiré rapport
                                flacons 57gr chacun ça fait peut chère Mondial relay offre
                                Livraison rapide Bons produit bien emballé
```

```
| prix | raisonnables
| Prix | très attractif trouve peut frais livraison élevé
| Prix | intéressant
| prix | vente c'est peu léger surtout harnais
| prix | bien plus raisonnable Bien cordialement
| prix | raisonnables
```

## 9 Analyse du sentiment

```
knitr::opts_chunk$set(  
  echo = TRUE,  
  message = FALSE,  
  warning = FALSE)  
library(readxl)  
library(tidyverse)  
library(quanteda)  
library(quanteda.textstats)  
library(quanteda.textplots)  
library(RColorBrewer)  
display.brewer.all()
```

## 10 Les données

```
data <- read_csv("data/data_trustpilot_oiseaux.csv")
```

# 11 *Sentiment analysis*

On va réaliser une analyse du sentiment du corpus. Pour cela, on utilise le dictionnaire des sentiments et émotions NRC, car il est disponible dans 40 langues, dont le français. Il existe d'autres dictionnaires de sentiments (positif vs négatif), par exemple AFINN ou BING, ainsi que des dictionnaires thématiques (LIWC par exemple), mais ils sont en anglais ou payants, donc utilisables pour des corpus en anglais ou lorsqu'on dispose d'un budget. Les dernières évolutions du traitement en langage naturel des IA (transformers et autres, dont ChatGPT est un exemple), permettent d'autres approches très pertinentes, en utilisant le *machine learning*, mais cela va au-delà des objectifs de ce cours.

Ici, on utilise le dictionnaire NRC à travers le package *syuzhet*. La fonction *get\_nrc\_sentiment* prend en entrée un vecteur de type caractère.

```
library(syuzhet)

# d<-get_nrc_sentiment(data$comments, language = "french")
# write_rds(d, "sentiment_trustpilot_oiseaux.rds")

d<-read_rds("data/sentiment_trustpilot_oiseaux.rds")
summary(d, digits=1)
```

anger	anticipation	disgust	fear	joy
Min. :0.0	Min. : 0.0	Min. :0.0	Min. :0.0	Min. :0.0
1st Qu.:0.0	1st Qu.: 0.0	1st Qu.:0.0	1st Qu.:0.0	1st Qu.:0.0
Median :0.0	Median : 1.0	Median :0.0	Median :0.0	Median :0.0
Mean :0.1	Mean : 0.9	Mean :0.1	Mean :0.1	Mean :0.6
3rd Qu.:0.0	3rd Qu.: 1.0	3rd Qu.:0.0	3rd Qu.:0.0	3rd Qu.:1.0
Max. :7.0	Max. :10.0	Max. :7.0	Max. :6.0	Max. :6.0
sadness	surprise	trust	negative	positive
Min. :0.0	Min. :0.0	Min. :0.0	Min. : 0.0	Min. : 0
1st Qu.:0.0	1st Qu.:0.0	1st Qu.:0.0	1st Qu.: 0.0	1st Qu.: 1
Median :0.0	Median :1.0	Median :1.0	Median : 0.0	Median : 2
Mean :0.2	Mean :0.7	Mean :0.9	Mean : 0.4	Mean : 2
3rd Qu.:0.0	3rd Qu.:1.0	3rd Qu.:1.0	3rd Qu.: 0.0	3rd Qu.: 3
Max. :7.0	Max. :5.0	Max. :8.0	Max. :12.0	Max. :14

```
data<-cbind(data,d)
```

```
data[600,8:17]
```

```
      anger anticipation disgust fear joy sadness surprise trust negative
600      0              1      0   1   1      0          1      1          0
      positive
600      1
```

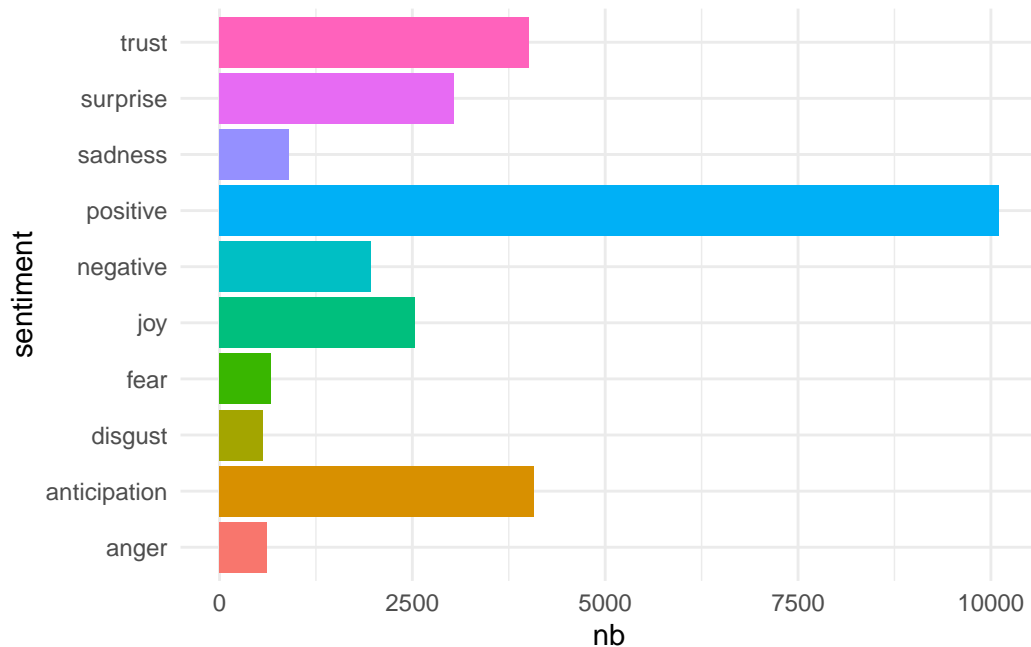
```
data[600,"comments"]
```

```
[1] "Commande reçu dans les temps pas de surprise. Très bon site"
```

Le dictionnaire comprend 10 variables, 8 émotions et 2 sentiments. Pour représenter les données, nous avons besoin de les transformer.

```
e<-d%>%
  pivot_longer(everything(),names_to = "sentiment", values_to = "nb")

ggplot(e, aes(sentiment, nb))+
  geom_col(aes(fill=sentiment),show.legend = FALSE)+
  theme_minimal()+
  coord_flip()
```



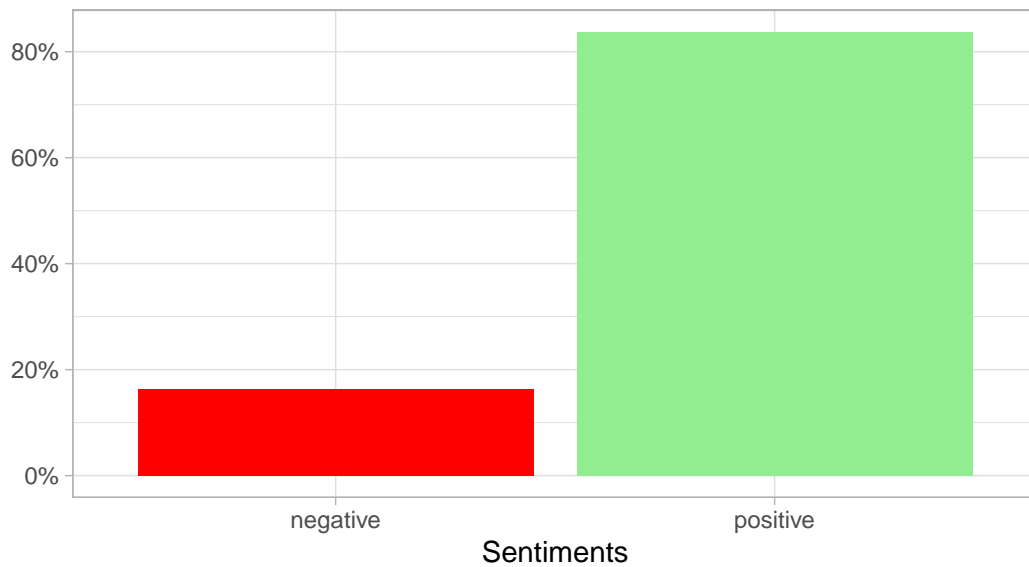
## 11.1 Les sentiments

Intéressons-nous d'abord aux sentiments :

```
sent<-d%>%
  select(positive,negative)%>%
  pivot_longer(everything(), names_to = "sentiment", values_to = "nb")%>%
  summarise(nb=sum(nb), .by = sentiment)%>%
  mutate(prop=nb/sum(nb))

ggplot(data=sent, aes(x=sentiment, y=prop)) +
  geom_bar(stat="identity", aes(fill=sentiment), show.legend = FALSE)+
  scale_y_continuous(labels=scales::percent) +
  labs(title = "Répartition des sentiments dans le corpus Oiseaux Mania",caption = "Données ")
  scale_fill_manual(values=c("red", "lightgreen"))+
  theme_light()
```

## Répartition des sentiments dans le corpus Oiseaux Mania



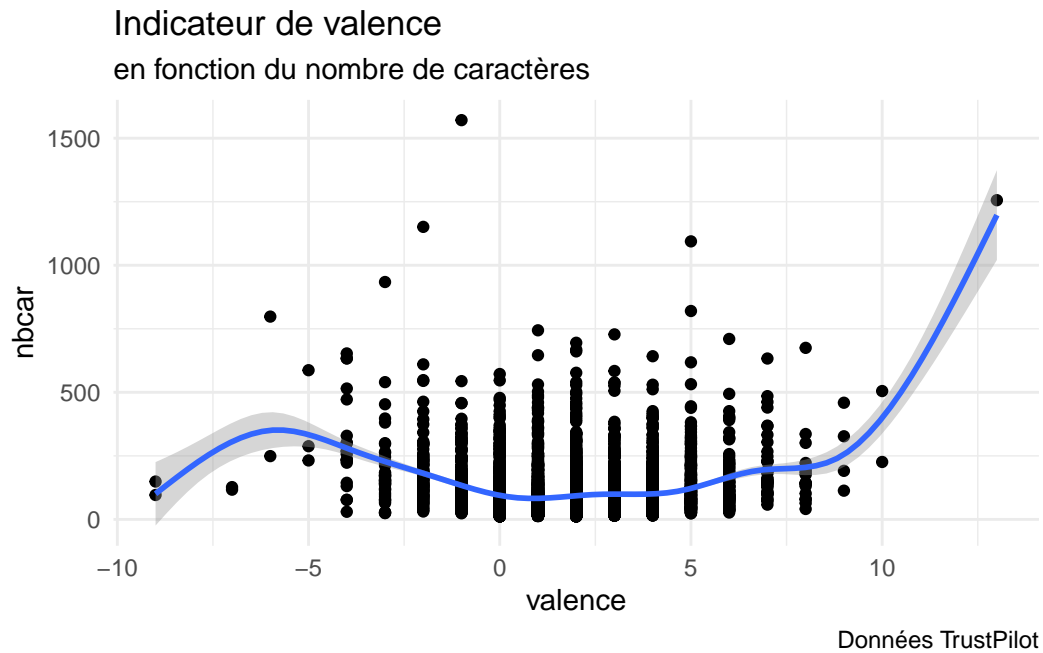
Données TrustPilot

Le corpus est très largement positif, ce qui n'est pas étonnant. On peut aussi créer d'autres indicateurs, comme la valence (différence positif-négatif) ou l'expressivité (somme de positif+négatif).

**Exercice :** personnalisez le graphique ci-dessous pour la variable d'expressivité.

```
data<-data%>%
  mutate(nbcar=nchar(comments),
         valence = positive-negative)

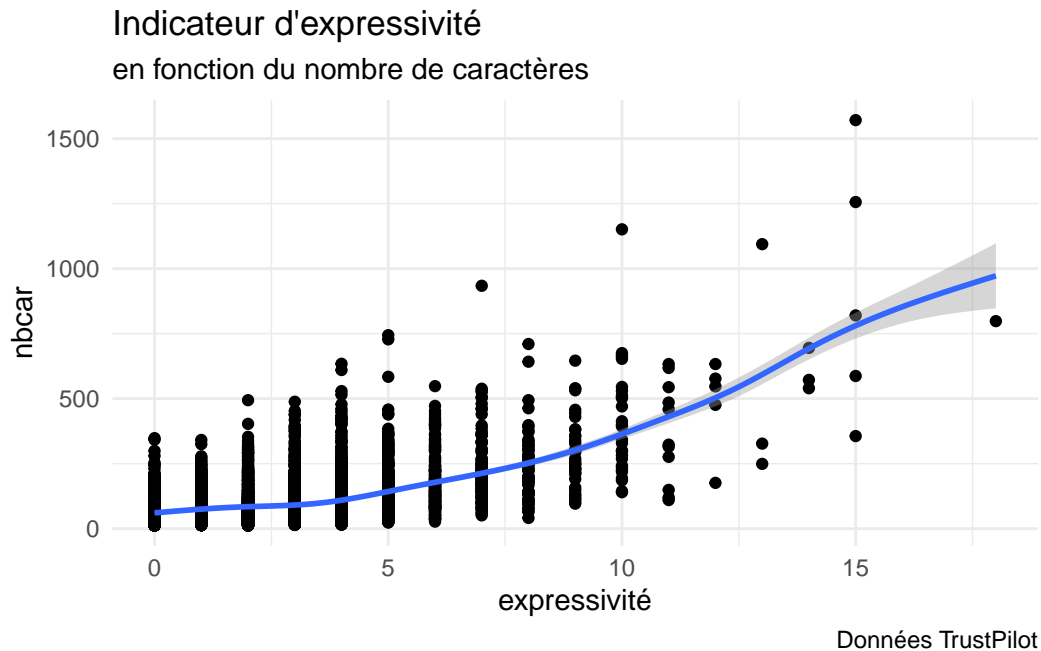
ggplot(data = data, aes(x = valence, y= nbcar))+
  geom_point()+
  geom_smooth()+
  labs(title="Indicateur de valence", subtitle = "en fonction du nombre de caractères", capt.
  theme_minimal()
```



```
data<-data%>%
  mutate(nbcar=nchar(comments),
         expressivité = positive+negative)

ggplot(data = data, aes(x = expressivité, y= nbcar))+
  geom_point()+
  geom_smooth()+
  labs(title="Indicateur d'expressivité", subtitle = "en fonction du nombre de caractères",
  theme_minimal()
```





## 11.2 Les émotions

Regardons maintenant ce qu'il en est de la répartition des émotions :

```
#On crée d'abord une palette pour les émotions
emocol<-c("yellow","chartreuse","olivedrab3","green4","royalblue3","purple3","red3","oranger")

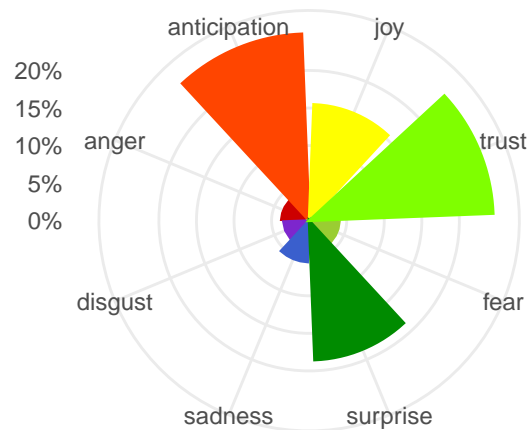
emo<-d%>%
  select(-positive, -negative)%>% #On récupère les émotions
  pivot_longer(everything(), names_to = "emotion", values_to = "nb")#On transforme le tableau

emo2<-emo%>%
  summarise(nb=sum(nb), .by = emotion)%>%
  mutate(prop=nb/sum(nb),
         emotion=factor(emotion, ordered = TRUE, levels = c("joy","trust","fear","surprise",

#On crée un graphique circulaire
ggplot(data=emo2, aes(x=emotion, y=prop, colour=emotion)) +
  geom_bar(stat="identity", aes(fill=emotion), show.legend = FALSE)+
  scale_y_continuous(labels=scales::percent)+
```

```
labs(title="Distribution des émotions \n dans le corpus Oiseaux Mania", caption="Données T
coord_polar()+
scale_color_manual(values=emocol)+ scale_fill_manual(values=emocol)+
theme_minimal()
```

### Distribution des émotions dans le corpus Oiseaux Mania

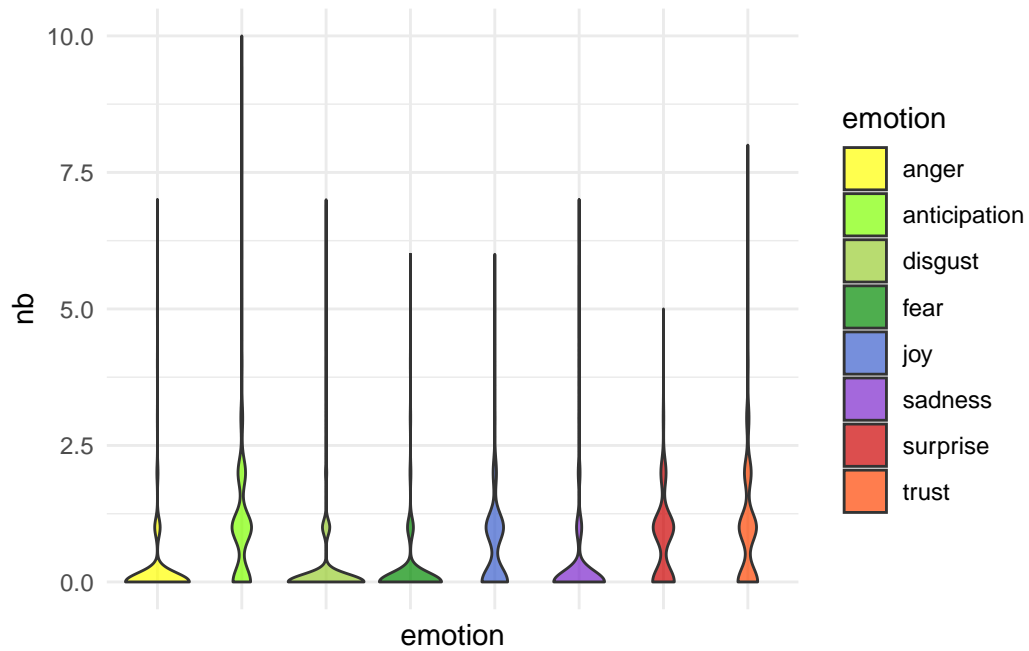


Emotions

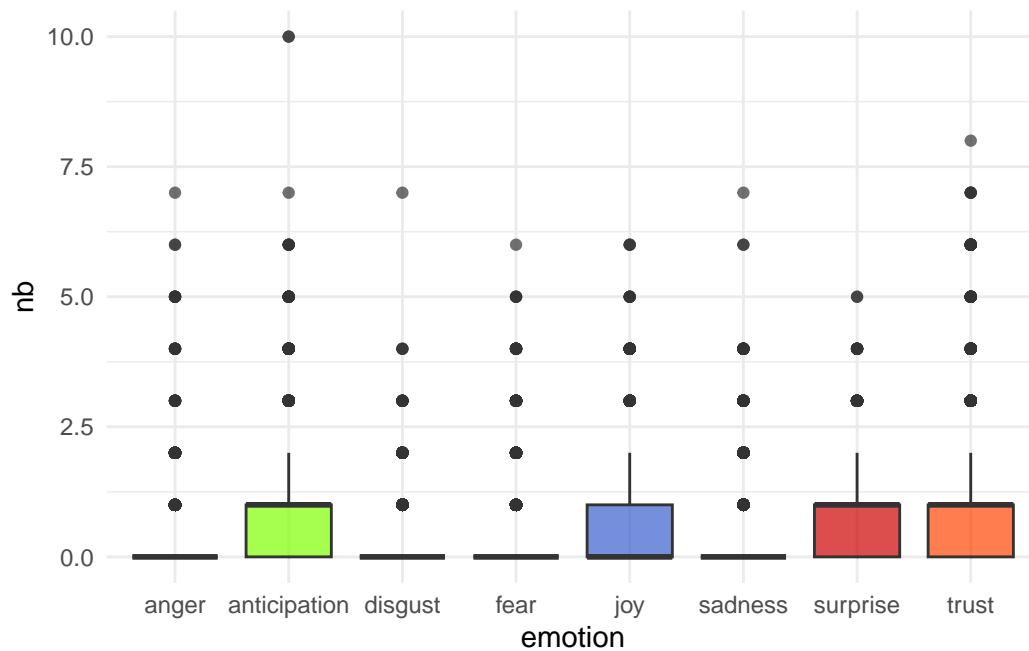
Données TrustPilot

#On regarde la répartition des émotions dans le corpus :

```
ggplot(emo, aes(x=emotion, y=nb))+
  geom_violin(aes(fill=emotion), alpha=0.7,adjust = 2)+
  theme_minimal()+ scale_fill_manual(values=emocol)+
  scale_x_discrete(labels=NULL)
```



```
ggplot(emo, aes(x=emotion, y=nb))+
  geom_boxplot(aes(fill=emotion,), alpha=0.7, adjust = 2, show.legend = FALSE)+
  theme_minimal()+ scale_fill_manual(values=emocol)
```



## 12 Évolution du corpus dans le temps

On va regarder comment les sentiments évoluent dans le temps. On doit tout d'abord créer une variable temporel dans notre jeu de données. Nous en avons déjà une, qui indique la date et l'heure à laquelle le commentaire a été posté. Nous allons la transformer pour regrouper les commentaires en fonction de l'année (on peut le faire pour les jours, les mois, les minutes, ...).

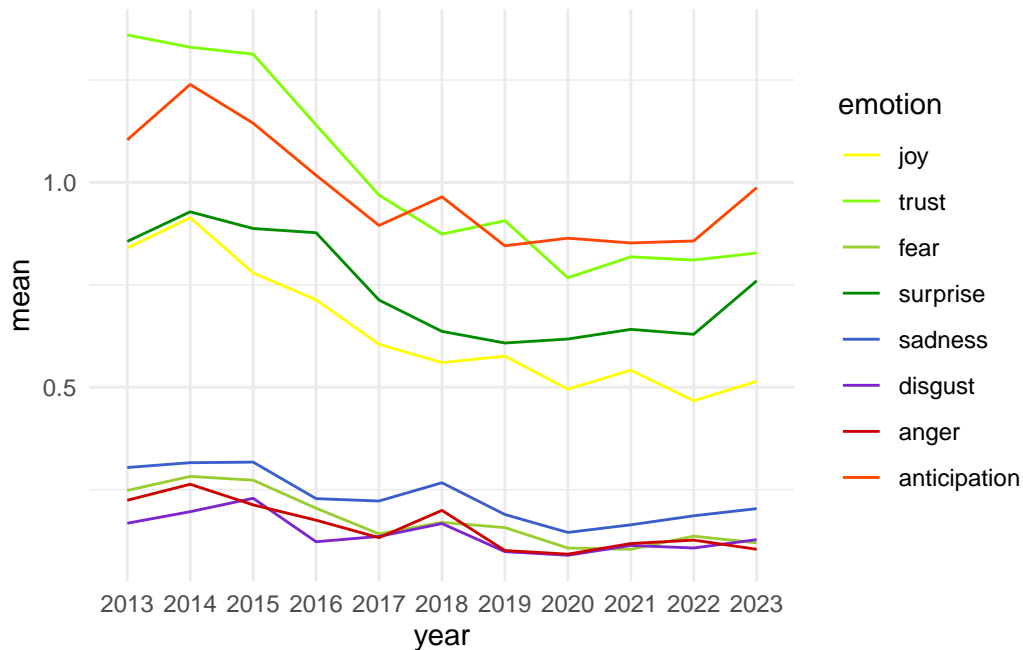
```
data%>%group_by(year)%>%summarise('nb com'=n())
```

```
# A tibble: 11 x 2
  year `nb com`
  <dbl>   <int>
1  2013     125
2  2014     209
3  2015     249
4  2016     171
5  2017     324
6  2018     341
7  2019     375
8  2020     735
9  2021     853
10 2022     623
11 2023     383
```

Regardons maintenant comment évolue les sentiments dans le temps :

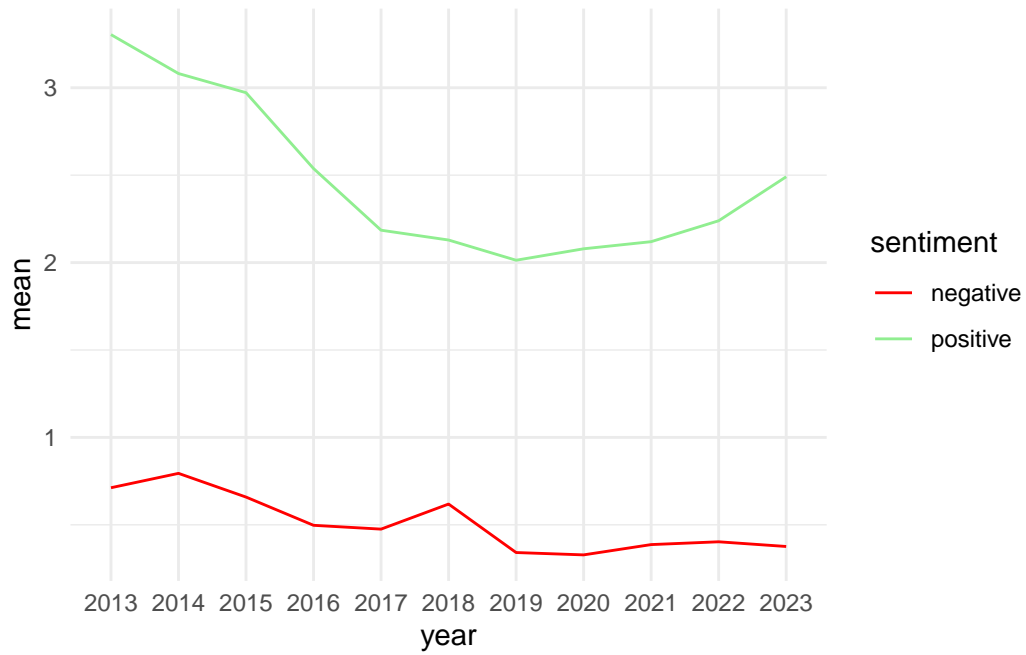
```
#les émotions
##mise en forme des données
gen_sent<-data%>%
  mutate(year=as.factor(year))%>%
  group_by(year)%>%
  summarise(across(7:14,~mean(.x, na.rm = T)))%>%
  na.exclude()%>%
  pivot_longer(-year, names_to = "emotion", values_to = "mean")%>%
  mutate(emotion=factor(emotion, ordered = TRUE,levels = c("joy","trust","fear","surprise",""
```

```
##graphique
ggplot(gen_sent,aes(x=year, y=mean,group=emotion)) +
  geom_line(aes(color=emotion), linewidth=0.5) +
  theme_minimal()+
  scale_color_manual(values = emocol)
```



```
#les sentiments
##mise en forme des données
gen_sent2<-data%>%
  mutate(year=as.factor(year))%>%
  group_by(year)%>%
  summarise(across(c(positive, negative),~mean(.x, na.rm = T)))%>%
  na.exclude()%>%
  pivot_longer(-year, names_to = "sentiment",values_to = "mean")

##graphique
ggplot(gen_sent2,aes(x=year, y=mean,group=sentiment)) +
  geom_line(aes(color=sentiment), linewidth=0.5) +
  theme_minimal()+
  scale_color_manual(values = c("red","lightgreen"))
```



Maintenant, on va s'intéresser aux mots.

# 13 Nuage de mots comparés

## 13.1 En fonction des années

On refait les manipulations préliminaires :

```
data<-data%>%
  mutate(year2=case_when(year<2017~"2013-2016",
                          year %in% c(2017:2019)~"2017-2019",
                          .default=as.character(year)))

corpus_oiseaux<-corpus(data, text_field = "comments")

tok<-tokens(corpus_oiseaux, remove_punct = TRUE, remove_numbers = TRUE, remove_symbols = TRUE,
             tokens_remove(stopwords("fr")))

dfm<-dfm(tok)
```

Comparons les mots en fonction des années :

```
dfmgrp<-dfm_group(dfm, groups = year2)
dfmgrp
```

Document-feature matrix of: 6 documents, 5,392 features (66.69% sparse) and 1 docvar.

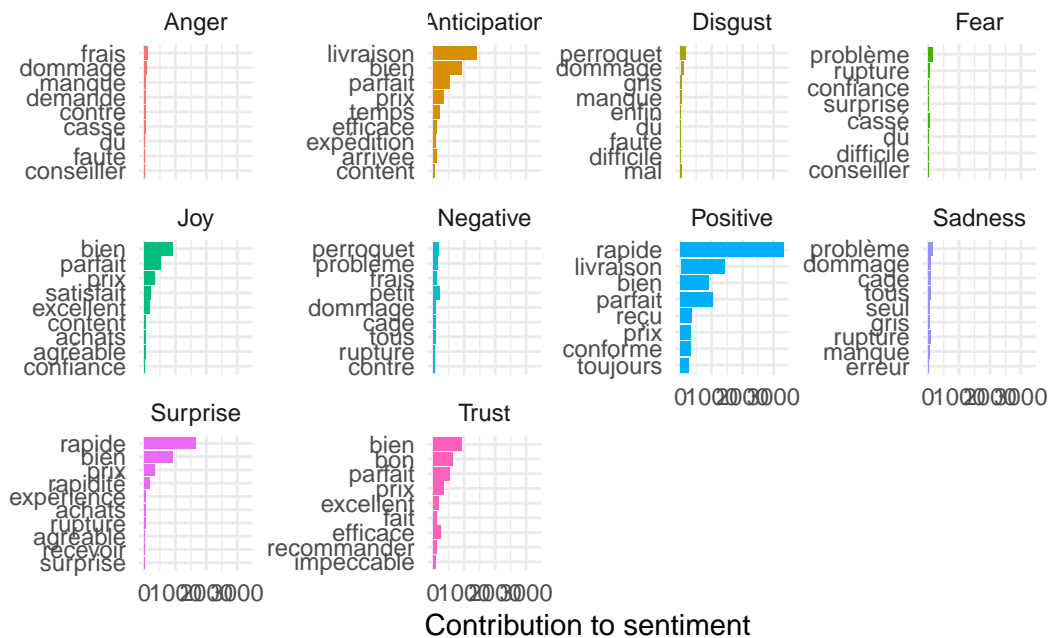
	features									
docs	comme	toujours	super	service	changez	rien	sauf	peut	être	là
2013-2016	38	26	93	58	0	71	6	17	9	8
2017-2019	32	61	74	78	0	76	5	13	24	4
2020	27	38	42	65	0	51	0	5	6	1
2021	35	49	49	64	2	62	8	11	11	2
2022	36	49	46	59	0	36	0	4	3	3
2023	16	51	22	27	2	21	3	6	3	4

[ reached max\_nfeat ... 5,382 more features ]





```
ggplot(sent_term, aes(reorder(word, value), value, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Contribution to sentiment",
       x = NULL) +
  coord_flip()+
  theme_minimal()
```



## **Part II**

### **Niveau 2 : Approfondissements**

## 14 Annotations et dépendances syntaxiques

```
library(readxl)
library(tidyverse)
library(quanteda)
library(quanteda.textstats)
library(quanteda.textplots)
library(RColorBrewer)
display.brewer.all()
```

## 15 Les données

```
data <- read_csv("data/data_trustpilot_oiseaux.csv")
```

Rows: 4388 Columns: 7

-- Column specification -----

Delimiter: ","

chr (4): auteur, date, month, comments

dbl (3): id, year, note

i Use `spec()` to retrieve the full column specification for this data.

i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

```
data$nb_caractere<-nchar(data$comments) #on compte le nombre de caractère de chaque commenta.  
summary(data$nb_caractere)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
12.0	41.0	74.0	104.3	130.0	1571.0

```
mean(data$nb_caractere)
```

```
[1] 104.2974
```

```
median(data$nb_caractere)
```

```
[1] 74
```

```
round(mean(data$nb_caractere),1)
```

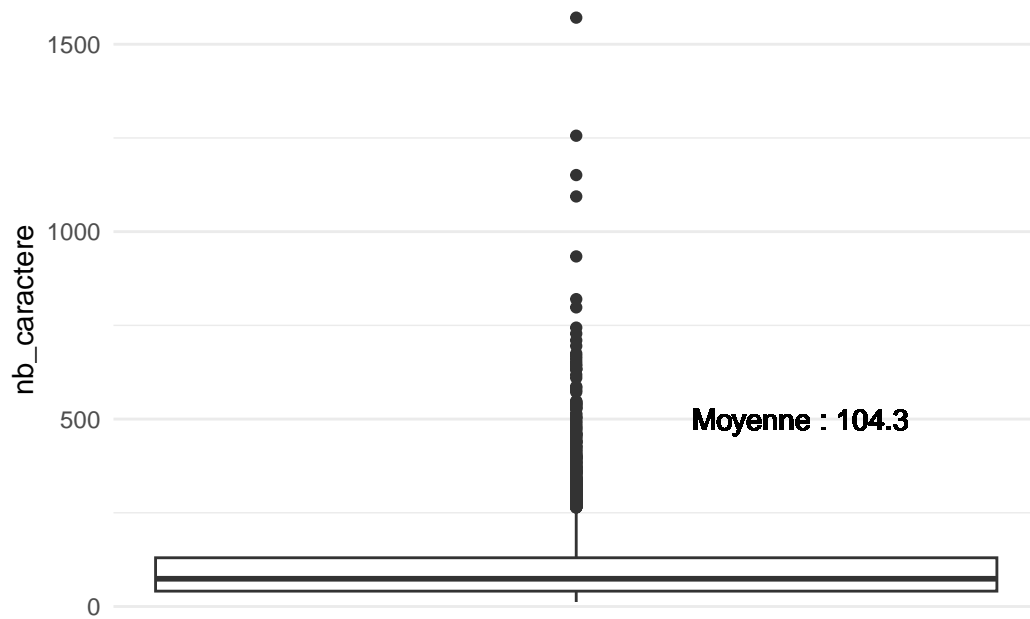
```
[1] 104.3
```

```

moy<-round(mean(na.omit(data$nb_caractere)), 1)

ggplot(data)+
  geom_boxplot(aes(nb_caractere))+
  geom_text(aes(x=500, y=0.2,label=paste("Moyenne :",moy)))+
  coord_flip()+
  scale_y_continuous(NULL, breaks = NULL)+
  theme_minimal()

```



## 16 Les traitements préliminaires

On reprend ce qu'on a fait au cours dernier, sans éliminer les termes trop fréquents :

```
corpus_oiseaux<-corpus(data, text_field = "comments")

tok<-tokens(corpus_oiseaux, remove_punct = TRUE, remove_numbers = TRUE, remove_symbols = TRUE)
  tokens_remove(stopwords("fr"))

dfm<-dfm(tok)
```

## 17 Co-occurrences

On va maintenant constituer des bi-grammes basés sur de nombreuses co-occurrences entre les termes :

```
# textstat_collocations(tok)
```

```
head(textstat_collocations(tok), 5)
```

	collocation	count	count_nested	length	lambda	z
1	livraison rapide	612	0	2	3.587755	56.29325
2	oiseaux mania	194	0	2	7.111407	38.50542
3	très bon	291	0	2	3.325458	37.28183
4	très bien	332	0	2	2.763473	37.07693
5	bonne qualité	116	0	2	4.440036	33.65412

```
tail(textstat_collocations(tok), 10)
```

	collocation	count	count_nested	length	lambda	z
4046	bien bien	3	0	2	-1.601302	-2.983435
4047	livraison produits	4	0	2	-1.539254	-3.248433
4048	site rapide	7	0	2	-1.205217	-3.274181
4049	commande commande	8	0	2	-1.154362	-3.338450
4050	bien rapide	10	0	2	-1.111673	-3.567851
4051	produits rapide	2	0	2	-2.293296	-3.616176
4052	livraison commande	9	0	2	-1.233460	-3.770310
4053	très a	2	0	2	-2.581564	-4.071675
4054	rapide rapide	4	0	2	-2.401230	-5.077424
4055	très très	22	0	2	-1.110720	-5.196198

```
colloc<-textstat_collocations(tok, min_count = 10, tolower = TRUE)
```

```
head(colloc, 10)
```

	collocation	count	count_nested	length	lambda	z
--	-------------	-------	--------------	--------	--------	---

1	livraison rapide	612	0	2	3.587755	56.29325
2	oiseaux mania	194	0	2	7.111407	38.50542
3	très bon	291	0	2	3.325458	37.28183
4	très bien	332	0	2	2.763473	37.07693
5	bonne qualité	116	0	2	4.440036	33.65412
6	rien dire	93	0	2	5.576935	32.54402
7	bon produit	146	0	2	3.412030	32.15574
8	envoi rapide	137	0	2	3.741715	27.61231
9	produit conforme	94	0	2	3.613662	27.52989
10	très satisfaite	154	0	2	3.288031	27.11619

```
tail(colloc,10)
```

	collocation	count	count_nested	length	lambda	z
356	livraison merci	10	0	2	-0.4024193	-1.285159
357	commande produits	11	0	2	-0.3922655	-1.311659
358	rapide site	13	0	2	-0.4424632	-1.601325
359	rapide livraison	23	0	2	-0.3478837	-1.653038
360	bien commande	11	0	2	-0.5705858	-1.910948
361	rapide a	10	0	2	-0.6915725	-2.214619
362	a rapide	10	0	2	-0.8863541	-2.840453
363	commande rapide	21	0	2	-0.6532274	-2.978690
364	bien rapide	10	0	2	-1.1116732	-3.567851
365	très très	22	0	2	-1.1107203	-5.196198

```
tok_cooc<-tokens_compound(tok, pattern = colloc[colloc$z>6.97,],join = TRUE)
tok["text400"]
```

Tokens consisting of 1 document and 7 docvars.

text400 :

```
[1] "Excellent" "service" "livraison" "rapide"
```

```
tok_cooc["text400"]
```

Tokens consisting of 1 document and 7 docvars.

text400 :

```
[1] "Excellent_service" "livraison_rapide"
```

Analyse de fréquence et représentation graphique :



```
dfm_cooc<-dfm(tok_cooc)
```

```
dfm_cooc2<-dfm_trim(dfm_cooc, max_termfreq = 170)
```

```
head(textstat_frequency(dfm_cooc2),20)
```

	feature	frequency	rank	docfreq	group
1	prix	169	1	160	all
2	service	165	2	154	all
3	perroquet	165	2	150	all
4	oiseaux	165	2	145	all
5	top	163	5	149	all
6	qualité	160	6	153	all
7	oiseaux_mania	142	7	129	all
8	rapidement	139	8	137	all
9	tres	127	9	107	all
10	articles	124	10	115	all
11	rapidité	124	10	117	all
12	reçu	122	12	114	all
13	sérieux	121	13	114	all
14	car	120	14	111	all
15	temps	120	14	115	all
16	satisfait	115	16	108	all
17	problème	114	17	102	all
18	comme	114	17	108	all
19	très_satisfaite	111	19	107	all
20	graines	107	20	87	all

```
textplot_wordcloud(dfm_cooc2, max_words = 200, color = brewer.pal(6, "Set2"))
```



17	tout	245	17	224	all
18	j'ai	234	18	189	all
19	colis	207	19	176	all
20	très_satisfaite	202	20	195	all
21	c'est	196	21	168	all
22	toujours	188	22	163	all
23	très_rapide	174	23	168	all
24	prix	169	24	160	all
25	service	165	25	154	all

```
dfm_cooc2<-dfm_trim(dfm_cooc, max_termfreq = 175)

textplot_wordcloud(dfm_cooc2, max_words = 100, color = brewer.pal(6, "Set2"))
```



## 17.1 Représentation en réseau des termes co-occurents

```
#
# fcm_cooc<-fcm(dfm_cooc2)
# fcm_cooc
# topfeatures(fcm_cooc)
```

```

# dim(fcm_cooc)
#
# feat<-names(topfeatures(fcm_cooc, 50))
# fcm_cooc_select<-fcm_select(fcm_cooc, pattern = feat, selection = "keep")
# dim(fcm_cooc_select)
#
# textplot_network(fcm_cooc_select, min_freq = 0.8, edge_color = "red" , edge_alpha = 0.5, v
#
#
# tpfeat<-tibble(feat=names(topfeatures(fcm_cooc,50)),n=topfeatures(fcm_cooc,50))
# tpfeat<-tpfeat%>%mutate(taille=n/150)
#
# textplot_network(fcm_cooc_select, min_freq = 0.8, edge_color = "red" , edge_alpha = 0.5, v
#
# textplot_network(fcm_cooc_select, min_freq = 0.8, edge_color = "red" , edge_alpha = 0.5, v
#
#
#
# textplot_network(fcm_cooc_select, min_freq = 0.8, edge_color = "red" , edge_alpha = 0.5, v

```

# 18 Annotations

Pour cette partie, on repart du jeu de données brut.

## 18.1 Détecter les langues

Dans le cas d'un corpus composé de plusieurs langues (par exemple, un corpus extrait de twitter), il peut être intéressant de filtrer le corpus à partir de la langue. On utilise un algorithme, qui peut être long à exécuter selon la taille du corpus, et qui est plutôt performant : cld3. Il repose sur un réseau de neurones développé par [Google](#)

```
library(cld3)

data$language<-detect_language(data$comments)
# data$language

data_fr<-data%>%filter(language=="fr")
```

## 18.2 POS

```
library(cleanNLP)

# cnlp_init_udpipe(model_name = "french")
#
# annotate<-cnlp_annotate(data$comments, verbose = 100)
# ann_token<-annotate$token
# write_csv2(ann_token, "annotation_oiseaux.csv")
# write_rds(ann_token, "annotation_oiseaux.rds")

ann_token<-read_rds("data/annotation_oiseaux.rds")

head(ann_token%>%filter(upos=="ADJ" | upos=="NOUN" | upos=="VERB"), 15)
```

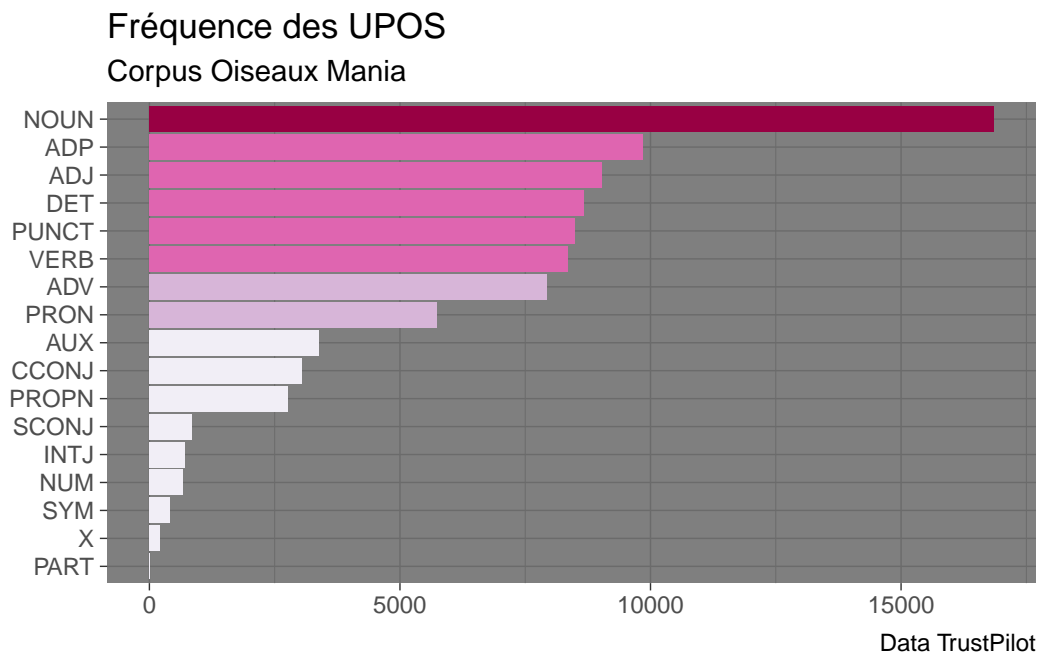
```
# A tibble: 15 x 11
  doc_id  sid tid  token  token_with_ws lemma  upos  xpos  feats tid_source
  <int> <int> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
1     1     1  1 4    super "super "    super ADJ  <NA>  Gend~ 5
2     1     1  1 5    service "service"  servi~ NOUN <NA>  Gend~ 0
3     1     2  2    changez "changez "  chang~ VERB <NA>  Mood~ 0
4     1     2  5    peut "peut "    pouvo~ VERB <NA>  Mood~ 2
5     1     2  8    cagnotte "cagnotte "  cagnot ADJ  <NA>  Gend~ 9
6     1     2  9    fidélité "fidélité "  fidél~ NOUN <NA>  Gend~ 5
7     1     2 11    est "est"      être  VERB <NA>  Mood~ 9
8     1     2 15    dirais "dirais "   dir   VERB <NA>  Mood~ 11
9     1     2 16    inutile "inutile "  inuti~ ADJ  <NA>  Gend~ 15
10    2     1  2    délai "délai "    délai NOUN <NA>  Gend~ 0
11    2     1  5    commande "commande "  comma~ NOUN <NA>  Gend~ 2
12    2     1  7    rapide "rapide "   rapide ADJ  <NA>  Gend~ 5
13    2     2  2    délais "délais "   délais NOUN <NA>  Gend~ 14
14    2     2  6    commande "commande "  comma~ NOUN <NA>  Gend~ 2
15    2     2  8    rapide "rapide"    rapide ADJ  <NA>  Gend~ 6
# i 1 more variable: relation <chr>
```

```
ann_token%>%filter(upos=="ADJ"|upos=="NOUN"|upos=="VERB")
```

```
# A tibble: 34,232 x 11
  doc_id  sid tid  token  token_with_ws lemma  upos  xpos  feats tid_source
  <int> <int> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
1     1     1  1 4    super "super "    super ADJ  <NA>  Gend~ 5
2     1     1  1 5    service "service"  servi~ NOUN <NA>  Gend~ 0
3     1     2  2    changez "changez "  chang~ VERB <NA>  Mood~ 0
4     1     2  5    peut "peut "    pouvo~ VERB <NA>  Mood~ 2
5     1     2  8    cagnotte "cagnotte "  cagnot ADJ  <NA>  Gend~ 9
6     1     2  9    fidélité "fidélité "  fidél~ NOUN <NA>  Gend~ 5
7     1     2 11    est "est"      être  VERB <NA>  Mood~ 9
8     1     2 15    dirais "dirais "   dir   VERB <NA>  Mood~ 11
9     1     2 16    inutile "inutile "  inuti~ ADJ  <NA>  Gend~ 15
10    2     1  2    délai "délai "    délai NOUN <NA>  Gend~ 0
# i 34,222 more rows
# i 1 more variable: relation <chr>
```

```
g<-ann_token%>%group_by(upos)%>%
  summarise(n=n())%>%
  filter(!is.na(upos))
```

```
ggplot(g)+
  geom_col(aes(reorder(upos,n),n, fill=n), show.legend = FALSE)+
  scale_fill_fermenter(palette = "PuRd", direction = 1)+
  coord_flip()+
  labs(title = "Fréquence des UPOS", subtitle = "Corpus Oiseaux Mania", caption = "Data TrustPilot")
  theme_dark()
```



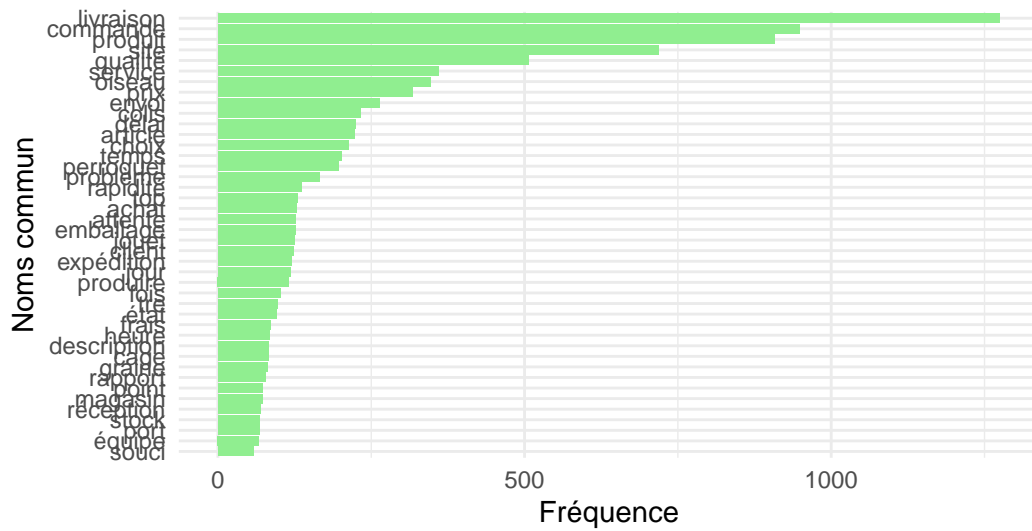
Maintenant, on va s'intéresser à des catégories grammaticales spécifiques :

```
vocab1<-ann_token%>%
  filter(upos=="NOUN")%>%
  summarise(freq=n(),.by=lemma)%>%
  filter(freq>55)

ggplot(vocab1,aes(x=reorder(lemma,freq),y=freq))+
  geom_bar(stat="identity",fill="lightgreen")+
  coord_flip()+
  theme_minimal()+
  labs(title = "Noms communs les plus fréquents",subtitle = "Corpus Oiseaux Mania", caption=
```

## Noms communs les plus fréquents

Corpus Oiseaux Mania

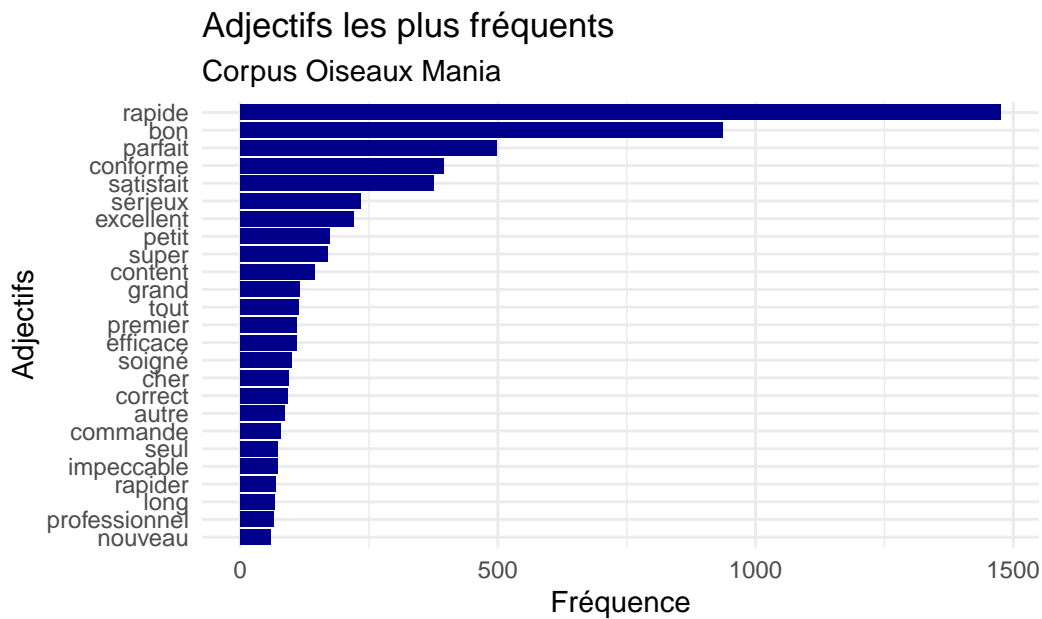


Data : TrustPilot

```
vocab1bis<-ann_token%>%
  filter(upos=="ADJ")%>%
  summarise(freq=n(),.by=lemma)%>%
  filter(freq>55)

ggplot(vocab1bis,aes(x=reorder(lemma,freq),y=freq))+
  geom_bar(stat="identity",fill="darkblue")+
  coord_flip()+
  theme_minimal()+
  labs(title = "Adjectifs les plus fréquents",subtitle = "Corpus Oiseaux Mania", caption="Data : TrustPilot")
```





```
vocab2<-ann_token%>%
  filter(upos=="NOUN" | upos=="VERB" | upos=="ADJ")%>%
  summarise(freq=n(),.by=c(lemma,upos))%>%
  filter(freq>30)%>%
  mutate(angle= 90 * sample(c(0, 1), n(), replace = TRUE, prob = c(75, 25)))

library(ggwordcloud)
ggplot(vocab2)+
  geom_text_wordcloud_area(aes(label=lemma, size=freq, color=freq, angle=angle))+
  scale_size_area(max_size = 24)+
  scale_color_fermenter(palette = "Set2")+
  theme_minimal()
```

Warning in wordcloud\_boxes(data\_points = points\_valid\_first, boxes = boxes, :  
Some words could not fit on page. They have been placed at their original  
positions.



```
ggplot(vocab2)+
  geom_text_wordcloud_area(aes(label=lemma, size=freq, color=upos, angle=angle))+
  scale_size_area(max_size = 24)+
  scale_color_manual(values=c("ADJ"="orange", "NOUN"="lightgreen", "VERB"="purple"))+
  theme_minimal()
```

Warning in wordcloud\_boxes(data\_points = points\_valid\_first, boxes = boxes, :  
One word could not fit on page. It has been placed at its original position.



## 19 Les dépendances syntaxiques

Quels sont les mots associés aux termes cibles ?

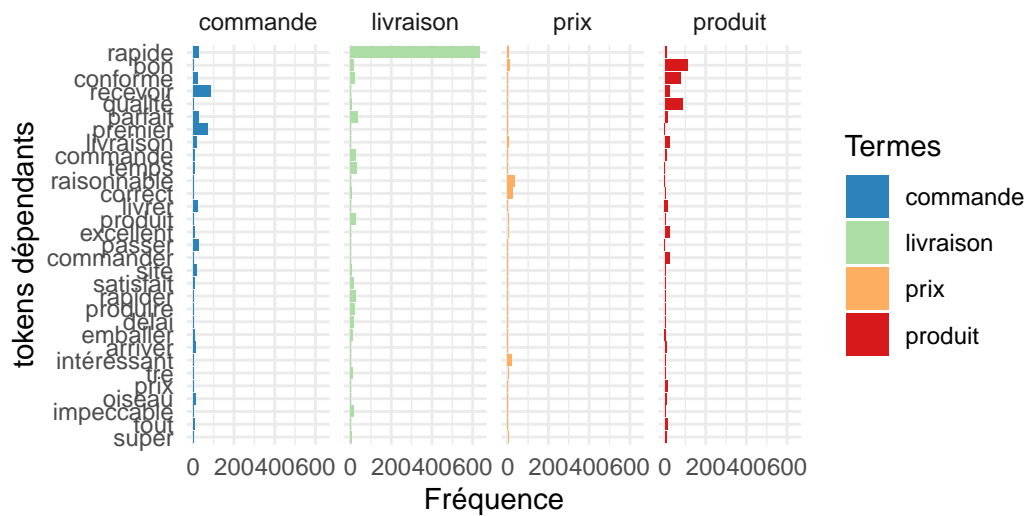
```
#on met à niveau la racine
ann_racine<- ann_token%>%
  left_join(ann_token,by= c("doc_id"="doc_id", "sid"="sid", "tid_source"="tid"), suffix=c("1", "2"))
#on filtre les relation nominales puis celle qui concerne les termes cibles
foo<-ann_racine %>%
  filter(relation == "amod"|relation == "acl"|relation == "nmod"|relation == "appos") %>%
  select(qual = lemma, source = lemma_source)%>%
  filter(source=="commande"|source=="livraison"|source=="produit"|source=="prix")%>%
  group_by(source,qual)%>%
  summarise(n=n())
```

`summarise()` has grouped output by 'source'. You can override using the  
`.groups` argument.

```
# On remet en forme les données
foo1<-foo%>%
  pivot_wider(names_from = source, values_from = n)%>%
  mutate(across(everything(), ~replace_na(.x,0)))%>%
  mutate(sum=rowSums(.,2:5))%>%
  filter(sum>15)%>%
  select(-sum)%>%
  pivot_longer(!qual, names_to = "source", values_to = "n")

ggplot(foo1,aes(x=reorder(qual,n), y=n, group=source))+
  geom_bar(stat="identity",aes(fill=source),position=position_dodge())+
  coord_flip()+
  scale_fill_brewer(palette="Spectral",direction = -1)+
  theme_minimal()+
  labs( title="Analyse des dépendances nominales", subtitle = "les termes du site et du service")
  facet_wrap(~source, ncol = 4)
```

Analyse des dépendances nominales  
les termes du site et du service



Data : TrustPilot sur Oiseaux Mania

## 20 *Topic Analysis*

```
library(readxl)
library(tidyverse)
library(quanteda)
library(quanteda.textstats)
library(quanteda.textplots)
library(RColorBrewer)
library(topicmodels)
library(ggwordcloud)
```

## 21 Les données

```
data <- read_csv("data/data_trustpilot_oiseaux.csv")
```

Rows: 4388 Columns: 7

-- Column specification -----

Delimiter: ","

chr (4): auteur, date, month, comments

dbl (3): id, year, note

i Use `spec()` to retrieve the full column specification for this data.

i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

## 22 Topics Analysis

On va maintenant s'intéresser à la détection et à l'analyse de topics. Il existe de nombreux algorithmes pour cela. On va en explorer un : le modèle LDA, pour Latent Dirichlet Allocation.

- Description du modèle LDA :

L'idée est la suivante : un corpus est considéré comme une collection de documents. Chaque document est considéré comme étant composé d'un mélange de topics. Chaque topic est considéré comme étant composé d'un mélange de tokens. L'algorithme calcule par itération les probabilités d'appartenance des tokens aux topics et des topics aux documents, ce qui nous permet de visualiser la composition des sujets identifiés.

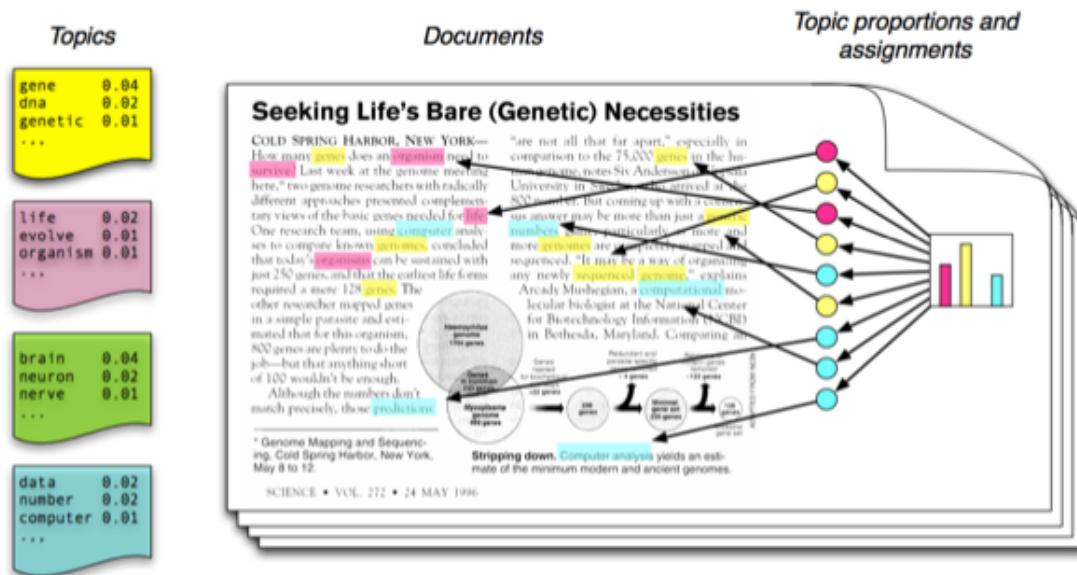


Figure source: Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55(4), 77-84.

Figure 22.1: Le modèle LDA



## 22.1 Le modèle LDA avec topicmodels

On travaille à partir du dfm. On doit transformer le format des données afin de l'injecter dans le modèle. On réduit le nombre de termes considérés, ce qui permet de réduire les temps de calcul et de trouver une solution convergente.

```
corpus_oiseaux<-corpus(data, text_field = "comments")

tok<-tokens(corpus_oiseaux, remove_punct = TRUE, remove_numbers = TRUE, remove_symbols = TRUE)
tokens_remove(stopwords("fr"))

dfm<-dfm(tok)

#On filtre les mots trop et trop peu fréquents
rem<-c("très","rapide","produit","livraison", "commande", "bien", "site", "a", "bon", "merci")

news_dfm <- dfm %>%
  dfm_remove(rem)%>%
  dfm_trim(min_termfreq = 0.8, termfreq_type = "quantile", # 80% des mots les plus fréquents
           max_docfreq = 0.2, docfreq_type = "prop")        #qui apparaissent dans max 20%

#On supprime les entrées vides
news_dfm <- news_dfm[ntoken(news_dfm) > 0,]

#On transforme en dtm, un format compris par le package topicmodels
dtm <- convert(news_dfm, to = "topicmodels")

#On lance le modèle
lda <- LDA(dtm, k = 5)

#On regarde les résultats
terms(lda,10)
```

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
[1,]	"produits"	"j'ai"	"super"	"produits"	"j'ai"
[2,]	"rien"	"emballé"	"qualité"	"qualité"	"rapidité"
[3,]	"rapidement"	"reçu"	"mania"	"satisfaite"	"oiseaux"
[4,]	"conforme"	"satisfaite"	"prix"	"colis"	"reçu"
[5,]	"service"	"toujours"	"bonne"	"reçu"	"c'est"
[6,]	"prix"	"colis"	"problème"	"toujours"	"colis"
[7,]	"oiseaux"	"produits"	"tout"	"rapidement"	"qualité"

```
[8,] "choix"      "plus"      "oiseaux"   "envoi"     "tout"
[9,] "satisfait" "envoi"     "tres"      "oiseaux"   "produits"
[10,] "temps"     "prix"      "chez"      "j'ai"      "service"
```

```
# topics(lda)
```

```
corpus_oiseaux["text996"]
```

Corpus consisting of 1 document and 6 docvars.

text996 :

"Seconde commande, satisfaction totale, expédition très rapid..."

```
corpus_oiseaux["text995"]
```

Corpus consisting of 1 document and 6 docvars.

text995 :

"Très bon produit. Livraison rapide et sérieux je recommande ..."

```
corpus_oiseaux["text999"]
```

Corpus consisting of 1 document and 6 docvars.

text999 :

"Très bon magasin emballage dans les normes on y trouve de to..."

## 22.2 Topic Analysis à partir de l'annotation des *part of speech*

Les résultats du modèle LDA sont très dépendants de la qualité du vocabulaire injecté. Plus on travaille ce vocabulaire, meilleurs sont les résultats. On va donc reprendre tout ce qu'on a fait jusqu'à présent pour améliorer les résultats de notre modèle : on récupère les annotations ; on filtre le vocabulaire pour ne garder que les noms, adjectifs et verbes ; on crée les collocations ; on filtre les occurrences trop et pas assez fréquentes.

```
ann_token<-read_rds("data/annotation_oiseaux.rds")
```

```
data<-ann_token%>%
```

```
  filter(upos=="NOUN"|upos=="VERB"|upos=="ADJ")%>%
```

```
  group_by(doc_id)%>%
```

```

summarise(text=paste(lemma,collapse = " "))%>%
inner_join(data, join_by("doc_id"=="id"))

corpus_new<-corpus(data, text_field = "text")
toks<-tokens(corpus_new)%>%
  tokens_replace(c("produire", "conformer","colir"), c("produit", "conforme","colis"))%>%
  tokens_remove(c(".",",",""))

colloc<-textstat_collocations(toks, min_count = 10, tolower = TRUE)

toks<-tokens_compound(toks, pattern = colloc[colloc$z>7,])

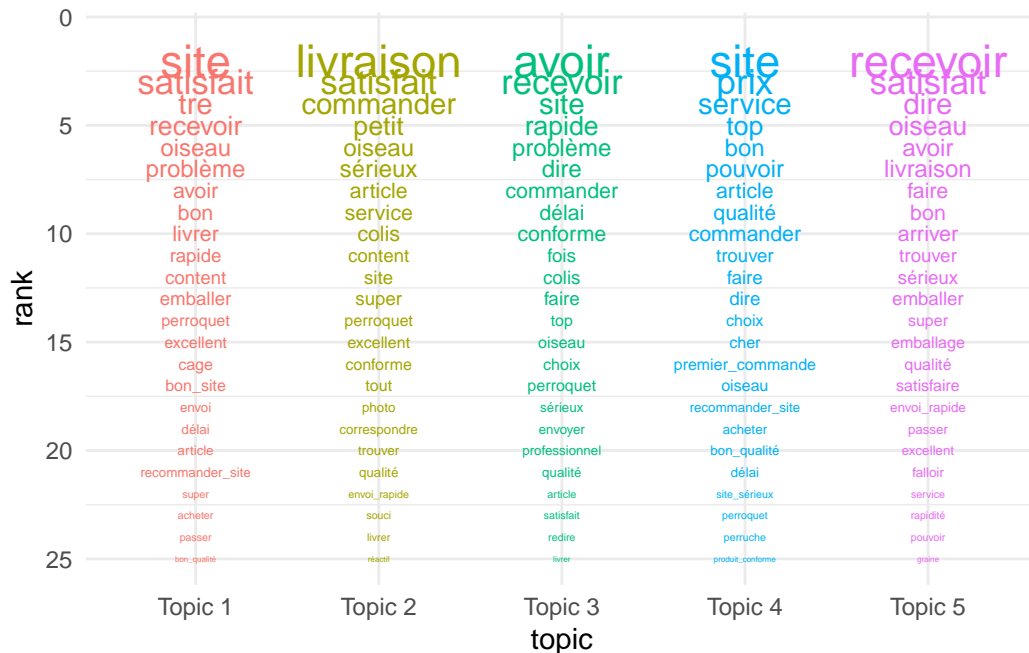
dfm_new<-dfm(toks)%>%
  dfm_trim(min_termfreq = 0.6, termfreq_type = "quantile",
           max_docfreq = 0.1, docfreq_type = "prop")
dtm_new <- convert(dfm_new, to = "topicmodels")

set.seed(1234)
lda <- LDA(dtm_new, k = 5)

term<-as_tibble(terms(lda,25))%>%
  mutate(rank=as.numeric(row.names(.)))%>%
  pivot_longer(-rank, names_to = "topic",values_to = "term")

ggplot(term, aes(x=topic, y= rank, group = term , label = term)) +
  scale_y_reverse() +
  geom_text(aes(color=topic,size=8/log(rank)))+
  theme_minimal()+
  scale_color_hue()+
  guides(color="none",size="none")

```



## 22.3 Déterminer le nombre de topics optimal

Le modèle LDA fonctionne à partir d'un nombre de topics donné. La question est donc de savoir quel est le nombre de topics optimal pour décrire notre corpus. Heureusement, des personnes ont créé des fonctions et des procédures pour nous aider dans cette quête. L'idée est de calculer différents modèles pour différents nombres de topics, et de comparer la qualité des résultats. La procédure ci-dessous est en deux parties :

- Tout d'abord, on compare la qualité de différents indicateurs sur un grand nombre de modèles, pour aboutir à une liste de quelques solutions à comparer plus en détail (de 3 à 10).
- Ensuite, on compare les résultats de la liste réduite de modèles, pour déterminer lequel a la meilleure distribution des topics entre les documents. La distribution recherchée est celle qui distingue le plus les documents en fonction des topics, tout en étant à droite de l'estimation d'une répartition uniforme des documents entre les topics. Le critère de parcimonie nous invite à choisir la solution avec le moins grand nombre de topics, en cas de résultats comparables.

```
##Etape 1 : les meilleures solutions
library(ldatuning)
library(magrittr)
```

Attachement du package : 'magrittr'

L'objet suivant est masqué depuis 'package:purrr':

set\_names

L'objet suivant est masqué depuis 'package:tidyr':

extract

```
result <- FindTopicsNumber(dtm_new,
  topics = c(seq(from = 2, to = 9, by = 1), seq(10, 25, 5)),
  metrics = c("Griffiths2004", "CaoJuan2009", "Arun2010", "Deveaud2014"),
  method = "Gibbs",
  control = list(seed = 0:4,
    nstart = 5,
    best = TRUE),
  mc.cores = 4L,
  verbose = TRUE
)
```

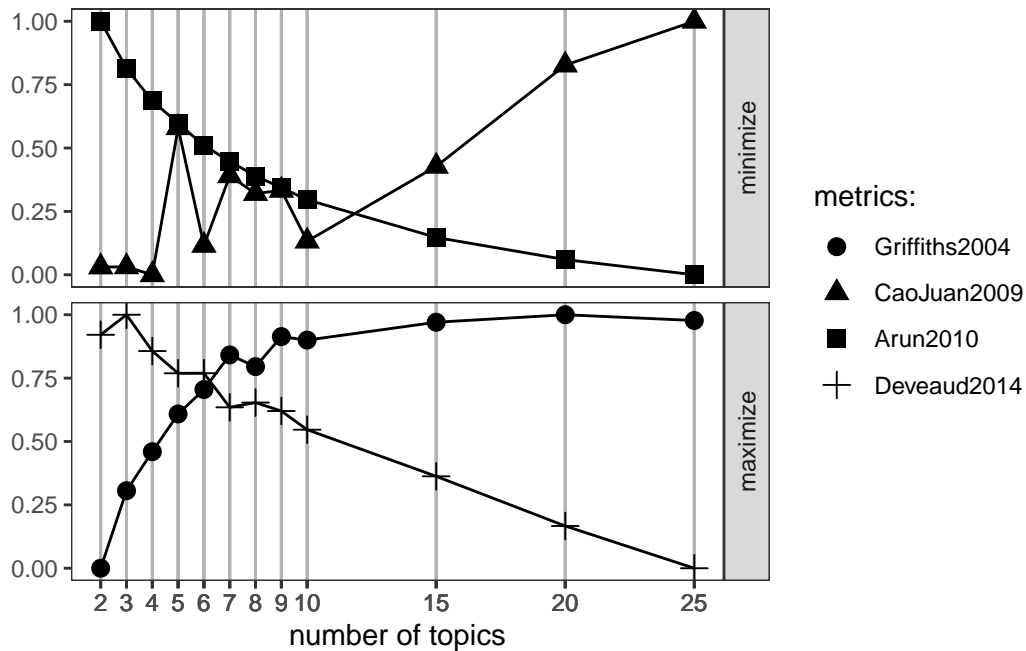
```
fit models... done.
calculate metrics:
  Griffiths2004... done.
  CaoJuan2009... done.
  Arun2010... done.
  Deveaud2014... done.
```

```
FindTopicsNumber_plot(result)
```

Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as of ggplot2 3.3.4.

i The deprecated feature was likely used in the ldatuning package.

Please report the issue at <<https://github.com/nikita-moor/ldatuning/issues>>.



```
##Etape 2 : comparaison des solutions
para <- tibble(k = c(6,7,8,9,10))
lemma_tm <- para %>%
  mutate(lda = map(k,
    function(k) LDA(
      k=k,
      x=dtm_new,
      method="Gibbs",
      control=list(seed = 0:4,
                    nstart = 5,
                    best = TRUE)
    )
  ))

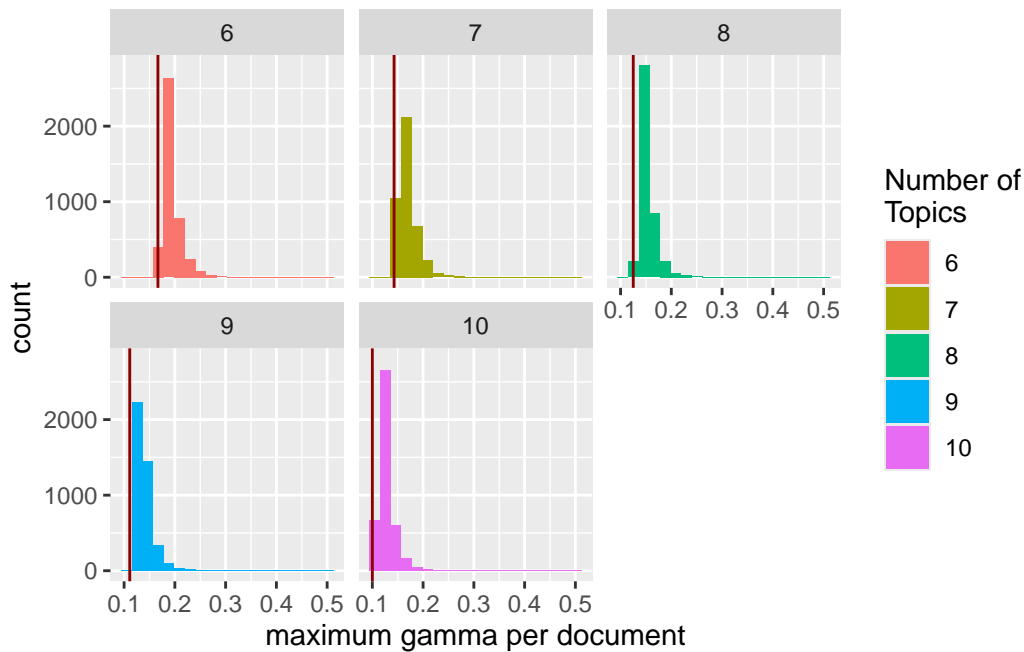
lemma_tm <- lemma_tm %>%
  mutate(lda_gamma = map(.x=lda,
    .f=tidytext::tidy,
    matrix="gamma"))

lemma_tm %>%
  unnest(lda_gamma) %>%
  group_by(k, document) %>%
  arrange(desc(gamma)) %>%
```

```

slice(1) %>%
ungroup() %>%
ggplot(aes(x=gamma, fill=factor(k))) +
geom_histogram(bins = 20) +
scale_fill_discrete(name = "Number of\nTopics") +
xlab("maximum gamma per document") +
facet_wrap(~k) +
geom_vline(aes(xintercept = 1/k),
            tibble(k=lemma_tm %$% unique(k)),
            color="darkred")

```



## 22.4 Représentation graphique

À partir de la solution retenue aux étapes précédentes, on va représenter les différents topics :

```

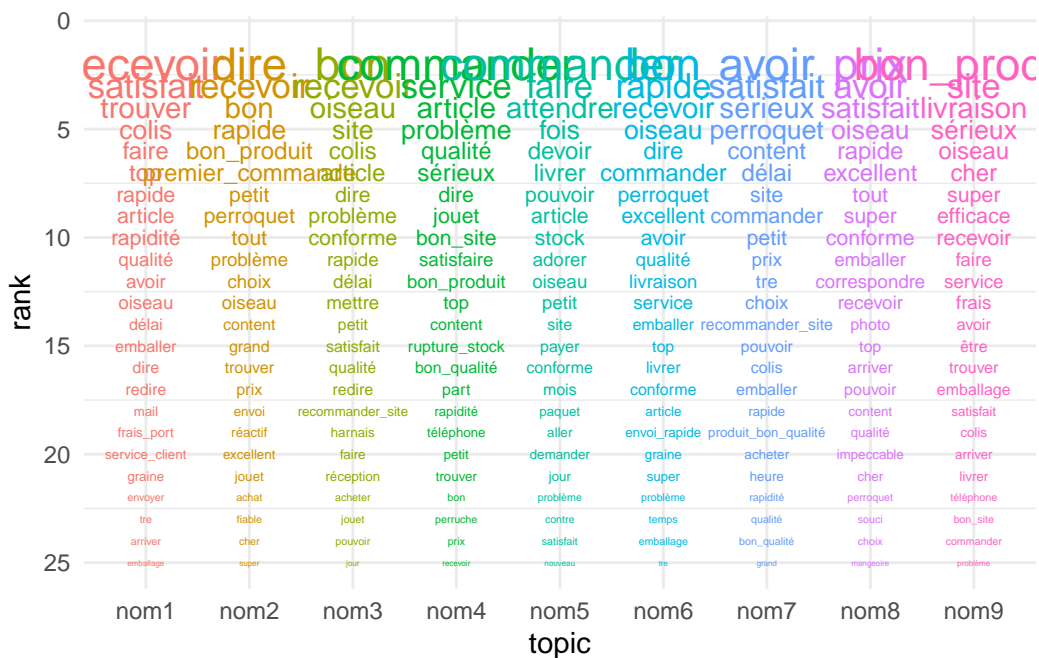
set.seed(1234)      #pour la répliquabilité des résultats
lda <- LDA(dtm_new, k = 9)

lda_res<-as.data.frame(terms(lda, 25))%>%
  rename(nom1='Topic 1',nom2='Topic 2', nom3='Topic 3', nom4='Topic 4', nom5='Topic 5',nom6=

```

```
mutate(rank=as.numeric(row.names(.)))%>%
pivot_longer(-rank, names_to = "topic", values_to = "term")

ggplot(lda_res, aes(x=topic, y= rank, group = term , label = term)) +
  scale_y_reverse() +
  geom_text(aes(color=topic,size=8/log(rank)))+
  theme_minimal()+
  scale_color_hue()+
  guides(color=FALSE,size=FALSE)
```





## 23 Theory-Driven LDA

Ici, on va forcer les *topics* grâce à la réalisation d'un dictionnaire. C'est utile quand on cherche à appliquer une théorie qui nous dit ce que l'on cherche à trouver. Par exemple, ici on s'intéresse aux attributs clés des logements oiseaux. Dans d'autre cas, on pourra chercher à expliquer les notes en fonction de *topics* qui reflètent les attributs clés. On peut réaliser le dictionnaire a priori ou après différentes analyses de *topics*, de co-occurrences, de fréquence, etc.

On commence par créer un dictionnaire.

```
dict<-dictionary(list(produit=c("produit*", "cage","oiseau","graine*"),
                        livraison=c("livr*","recepti*","délai"),
                        commande=c("command*","emballage","envoi*"),
                        site="*site*",
                        prix=c("*prix*","frais_port")
                      ))
dict
```

Dictionary object with 5 key entries.

```
- [produit]:
  - produit*, cage, oiseau, graine*
- [livraison]:
  - livr*, recepti*, délai
- [commande]:
  - command*, emballage, envoi*
- [site]:
  - *site*
- [prix]:
  - *prix*, frais_port
```

```
head(dfm_lookup(dfm_new,dict))
```

Document-feature matrix of: 6 documents, 5 features (73.33% sparse) and 6 docvars.

	features					
docs	produit	livraison	commande	site	prix	
1	1	0	0	0	0	0

2	0	2	1	0	0
3	2	1	0	0	0
4	0	0	1	0	0
5	0	1	0	0	0
6	0	1	0	1	0

On utilise ensuite le package 'seededlda' pour lancer le modèle semi-supervisé.

```
library(seededlda)
```

Le chargement a nécessité le package : proxyC

Attachement du package : 'proxyC'

L'objet suivant est masqué depuis 'package:stats':

```
dist
```

Attachement du package : 'seededlda'

Les objets suivants sont masqués depuis 'package:topicmodels':

```
terms, topics
```

L'objet suivant est masqué depuis 'package:stats':

```
terms
```

```
set.seed(1234)
slda<-textmodel_seededlda(dfm_new, dict, residual = T)
terms(slda,20)
```

	produit	livraison
[1,]	"oiseau"	"livraison"
[2,]	"cage"	"livrer"
[3,]	"graine"	"délai"
[4,]	"perroquet"	"dire"

[5,]	"produit_qualité"	"bon_produit"
[6,]	"produit_conforme"	"satisfaire"
[7,]	"avoir"	"problème"
[8,]	"content"	"satisfait"
[9,]	"trouver"	"service"
[10,]	"petit"	"bon"
[11,]	"jouet"	"top"
[12,]	"acheter"	"livraison_rapide_produit"
[13,]	"harnais"	"livraison_rapide_recommander"
[14,]	"perruche"	"temps"
[15,]	"produit_bon_qualité"	"cher"
[16,]	"grand"	"impeccable"
[17,]	"produit_conforme_description"	"emballer"
[18,]	"tout"	"livreur"
[19,]	"passer"	"livraison_temps"
[20,]	"arriver"	"correct"
	commande	site
[1,]	"commander"	"site"
[2,]	"recevoir"	"rapide"
[3,]	"envoi_rapide"	"bon_site"
[4,]	"emballage"	"recommander_site"
[5,]	"envoi"	"sérieux"
[6,]	"satisfait"	"site_sérieux"
[7,]	"article"	"tre"
[8,]	"conforme"	"satisfait"
[9,]	"commande_recevoir"	"siter"
[10,]	"emballer"	"redire"
[11,]	"envoi_rapide_soigné"	"rapider"
[12,]	"commande_arriver"	"premier_commande"
[13,]	"commande_passer"	"super_site"
[14,]	"correspondre"	"super"
[15,]	"problème"	"choix"
[16,]	"content"	"autre_site"
[17,]	"commande_livrer"	"conseiller"
[18,]	"rapidité"	"bon_produit"
[19,]	"rapide"	"achat"
[20,]	"envoie"	"hésiter"
	other	prix
[1,]	"avoir"	"prix"
[2,]	"faire"	"bon"
[3,]	"colis"	"excellent"
[4,]	"pouvoir"	"qualité"
[5,]	"jour"	"service"
		"frais_port"
		"top"
		"article"
		"magasin"
		"prix_raisonnable"
		"satisfait"
		"professionnel"
		"super"
		"équipe"
		"réactif"
		"téléphone"
		"bon_rapport_qualité_prix"
		"prix_correct"
		"trouver"
		"expédition_rapide"

[6,] "fois"  
[7,] "recevoir"  
[8,] "attendre"  
[9,] "dire"  
[10,] "autre"  
[11,] "aller"  
[12,] "part"  
[13,] "être"  
[14,] "petit"  
[15,] "devoir"  
[16,] "problème"  
[17,] "mettre"  
[18,] "attente"  
[19,] "voir"  
[20,] "mail"

## 24 Expliquer les notes

Dans cette dernière partie, nous allons nous intéresser aux notes et tenter de les expliquer à l'aide de l'analyse de *topics*.

### 24.1 NPS

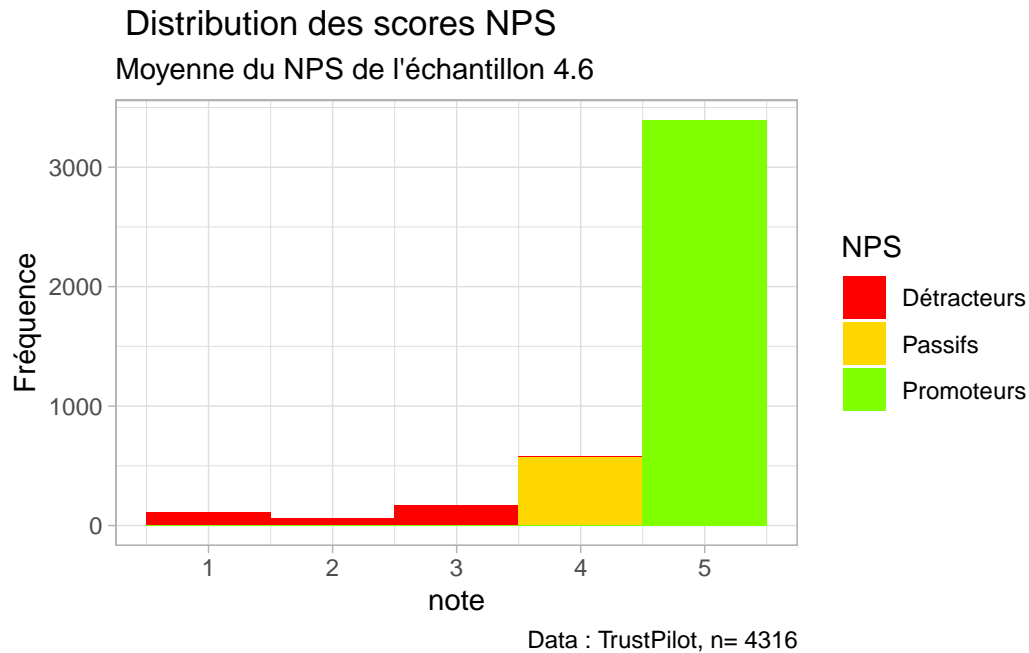
Dans un premier temps, nous allons regarder le Net Promoter Score (NPS), puis nous étudierons les discours des promoteurs, détracteurs et passifs.

Tout d'abord, nous créons nos catégories en fonction des notes.

```
col<- c("red","gold", "chartreuse")

data<-data %>%
  mutate(NPS=case_when(note<4~"Détracteurs",
                        note==4~"Passifs",
                        note>4~"Promoteurs"))

ggplot(data, aes(x=note))+
  geom_histogram(binwidth = 1, aes(fill=NPS))+
  labs( title= " Distribution des scores NPS",
        subtitle = paste("Moyenne du NPS de l'échantillon",round(mean(data$note),1)),
        caption = paste("Data : TrustPilot, n=",nrow(data)),
        y = "Fréquence")+
  scale_fill_manual(values=col)+
  theme_light()
```



Puis nous réalisons un nuage de mots pour chaque groupe, afin d'avoir une idée de ce qui est exprimé.

```
dfm_new$NPS<-data$NPS
# docvars(toks)

dfm_gp <-dfm_new %>%
  dfm_group(groups = NPS)
# dfm_gp

stat<- dfm_gp %>%
  textstat_frequency(n = 30, groups = NPS)
stat
```

	feature	frequency	rank	docfreq	group
1	avoir	103	1	1	Détracteurs
2	commander	81	2	1	Détracteurs
3	recevoir	79	3	1	Détracteurs
4	livraison	77	4	1	Détracteurs
5	colis	41	5	1	Détracteurs
6	site	40	6	1	Détracteurs
7	pouvoir	38	7	1	Détracteurs
8	article	38	7	1	Détracteurs

9	livrer	37	9	1 Détracteurs
10	faire	34	10	1 Détracteurs
11	fois	31	11	1 Détracteurs
12	problème	28	12	1 Détracteurs
13	dire	28	12	1 Détracteurs
14	attendre	28	12	1 Détracteurs
15	jour	28	12	1 Détracteurs
16	devoir	28	12	1 Détracteurs
17	cher	26	17	1 Détracteurs
18	graine	26	17	1 Détracteurs
19	autre	25	19	1 Détracteurs
20	envoyer	25	19	1 Détracteurs
21	payer	25	19	1 Détracteurs
22	acheter	25	19	1 Détracteurs
23	trouver	23	23	1 Détracteurs
24	cage	23	23	1 Détracteurs
25	long	23	23	1 Détracteurs
26	oiseau	21	26	1 Détracteurs
27	déçu	21	26	1 Détracteurs
28	mois	21	26	1 Détracteurs
29	réponse	20	29	1 Détracteurs
30	semaine	20	29	1 Détracteurs
31	avoir	85	1	1 Passifs
32	livraison	78	2	1 Passifs
33	commander	56	3	1 Passifs
34	site	53	4	1 Passifs
35	recevoir	50	5	1 Passifs
36	rapide	42	6	1 Passifs
37	satisfait	41	7	1 Passifs
38	bon	40	8	1 Passifs
39	petit	35	9	1 Passifs
40	pouvoir	34	10	1 Passifs
41	perroquet	34	10	1 Passifs
42	colis	33	12	1 Passifs
43	oiseau	32	13	1 Passifs
44	trouver	30	14	1 Passifs
45	faire	29	15	1 Passifs
46	article	28	16	1 Passifs
47	frais_port	28	16	1 Passifs
48	cher	26	18	1 Passifs
49	problème	23	19	1 Passifs
50	dire	23	19	1 Passifs
51	bon_produit	21	21	1 Passifs

52	livrer	21	21	1	Passifs
53	emballer	21	21	1	Passifs
54	délai	20	24	1	Passifs
55	satisfaire	20	24	1	Passifs
56	cage	20	24	1	Passifs
57	arriver	19	27	1	Passifs
58	prix	19	27	1	Passifs
59	emballage	19	27	1	Passifs
60	super	18	30	1	Passifs
61	rapide	295	1	1	Promoteurs
62	satisfait	284	2	1	Promoteurs
63	site	279	3	1	Promoteurs
64	livraison	275	4	1	Promoteurs
65	commander	269	5	1	Promoteurs
66	oiseau	228	6	1	Promoteurs
67	avoir	222	7	1	Promoteurs
68	recevoir	186	8	1	Promoteurs
69	bon	173	9	1	Promoteurs
70	sérieux	157	10	1	Promoteurs
71	dire	150	11	1	Promoteurs
72	qualité	146	12	1	Promoteurs
73	service	143	13	1	Promoteurs
74	top	141	14	1	Promoteurs
75	bon_produit	138	15	1	Promoteurs
76	excellent	138	15	1	Promoteurs
77	article	134	17	1	Promoteurs
78	super	133	18	1	Promoteurs
79	content	127	19	1	Promoteurs
80	perroquet	122	20	1	Promoteurs
81	trouver	119	21	1	Promoteurs
82	problème	117	22	1	Promoteurs
83	faire	111	23	1	Promoteurs
84	emballer	111	23	1	Promoteurs
85	conforme	108	25	1	Promoteurs
86	prix	99	26	1	Promoteurs
87	tre	93	27	1	Promoteurs
88	délai	92	28	1	Promoteurs
89	petit	92	28	1	Promoteurs
90	rapidité	88	30	1	Promoteurs

```
ggplot(stat, aes(label = feature)) +
  geom_text_wordcloud(aes(size=log(frequency), color=group)) +
```



```
theme_minimal()+
facet_wrap(vars(group))+
scale_color_manual(values=col)+
labs(title="Nuage des 30 mots les plus fréquents (Par groupes)",
      caption = "La taille des mots est proportionnelle au log de leurs fréquences")
```

Warning in wordcloud\_boxes(data\_points = points\_valid\_first, boxes = boxes, :  
Some words could not fit on page. They have been placed at their original  
positions.

## Nuage des 30 mots les plus fréquents (Par groupes)



La taille des mots est proportionnelle au log de leurs fréquences

Maintenant, nous nous intéressons à ce qui caractérise chacun des groupes par rapport aux autres, grâce à la mesure du *keyness*.

```
graph_promoteur<-textstat_keyness(dfm_gp, target = "Promoteurs")%>%
  textplot_keyness(n = 30L, labelsizes = 2, show_legend = FALSE,
                  show_reference = FALSE, color = c("Darkgreen", "gray"))+
  labs(x=NULL)

graph_detracteur <- textstat_keyness(dfm_gp, target = "Détracteurs" )%>%
  textplot_keyness(n = 30L, labelsizes = 2, show_legend = FALSE,
                  show_reference = FALSE, color = c("firebrick", "gray"))+
```

```

labs(x=NULL)

graph_passif <- textstat_keyness(dfm_gp, target = "Passifs")%>%
  textplot_keyness(n = 30L, labelsize = 2, show_legend = FALSE, show_reference = FALSE,
  labs(x=NULL)

library(cowplot)

```

Attachement du package : 'cowplot'

L'objet suivant est masqué depuis 'package:lubridate':

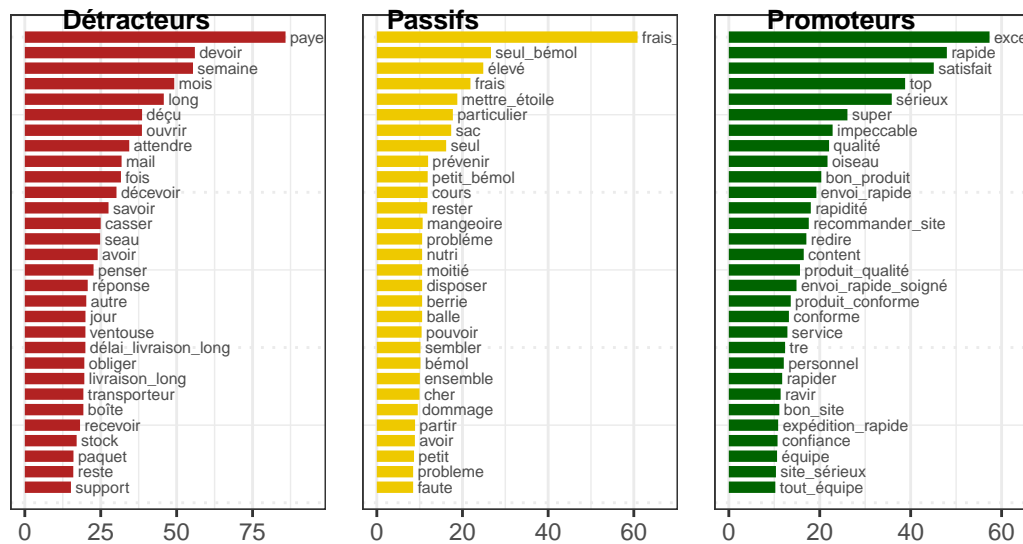
stamp

```

p<- plot_grid(graph_detracteur, graph_passif ,graph_promoteur, labels = c('Détracteurs', 'P
title <- ggdraw() + draw_label("NPS : Les raisons qui conduisent à la recommandation (keynes
note <- ggdraw()+ draw_text("Les valeurs représentent le keyness des termes.\nIl mesure leur
plot_grid(title, p,note, ncol=1, rel_heights=c(0.1, 1)) # rel_heights values control title m

```

## S : Les raisons qui conduisent à la recommandation (keynes)



Les valeurs représentent le keyness des termes.  
Il mesure leur caractère distinctif par une statistique du  $\chi^2$

## 24.2 En fonction des topics

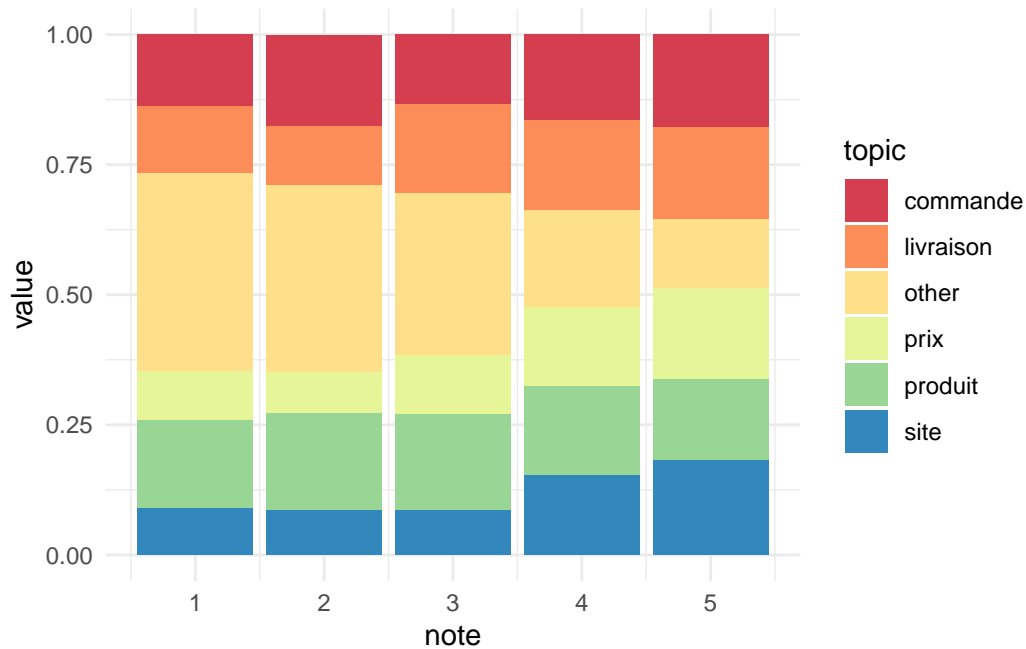
Maintenant, nous cherchons à voir la répartition des *topics* dans les notes, pour comprendre si certains *topics* contribuent plus ou moins à la satisfaction.

```
theta<-as.data.frame(slda$theta)%>%mutate(doc_id=as.numeric(row.names(.)))
data<-inner_join(data, theta)
```

Joining with `by = join\_by(doc\_id)`

```
foo<-data%>%select(note, produit, livraison, commande, site,prix, other)%>%
  pivot_longer(-note, names_to = "topic", values_to = "value")

ggplot(foo,aes(x=note, y=value, group=topic))+
  geom_bar(position="fill",stat="identity", aes(fill=topic))+
  scale_fill_brewer(palette="Spectral")+
  theme_minimal()
```



```
#Pour finir, une petite régression !
fit<-lm(note~produit+livraison+commande+site+prix, data =data)
summary(fit)
```

Call:

```
lm(formula = note ~ produit + livraison + commande + site + prix,
    data = data)
```

Residuals:

Min	1Q	Median	3Q	Max
-4.1270	-0.0206	0.1766	0.3033	1.9858

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	2.51256	0.07178	35.01	<2e-16 ***
produit	1.92260	0.11146	17.25	<2e-16 ***
livraison	2.45543	0.10907	22.51	<2e-16 ***
commande	2.44823	0.10885	22.49	<2e-16 ***
site	2.95498	0.10736	27.52	<2e-16 ***
prix	2.79509	0.10718	26.08	<2e-16 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7591 on 4310 degrees of freedom

Multiple R-squared: 0.1952, Adjusted R-squared: 0.1942

F-statistic: 209 on 5 and 4310 DF, p-value: < 2.2e-16

# A Webscraping

```
library(tidyverse)
library(scales)
library(quanteda)
library(quanteda.textstats)
library(httr)
library(rvest)
library(polite)

### Fichier TP_pets_website.csv

##### Scraping sites web
url_start<-"https://fr.trustpilot.com/categories/animals_pets"

trustpilot_website<-function(url_start){

  session <- bow(url_start)
  session$user_agent<-"Googlebot"
  message("Scraping ", url_start)
  page<-nod(session, url_start) %>%
    scrape(verbose=TRUE)
  i<-page%>%html_elements(".styles_paginationWrapper__fukEb")%>%
    html_element("a.button_button__T34Lr:nth-child(5)")%>%
    html_text()%>%as.numeric()
  website <- NULL

  for (j in 1:i){
    url<-paste0(url_start,"?page=",j)
    Sys.sleep(5)
    session <- bow(url)
    session$user_agent<-"Googlebot"
    message("Scraping ", url)
    page<-nod(session, url) %>%
      scrape(verbose=TRUE)
```

```

company_card <- page %>%
  html_elements("div.styles_wrapper__2J0o2:nth-of-type(n+4)")

website_name <- company_card %>%
  html_element("p.typography_heading-xs__jSwUz") %>%
  html_text()

nb_avis <- company_card %>%
  html_element("p.typography_body-m__xgxZ_") %>%
  html_text()

localisation <- company_card %>%
  html_element("span.styles_metadataItem__Qn_Q2") %>%
  html_text()

type <- company_card %>%
  html_element("div.styles_desktop__U5iWw") %>%
  html_text()

lien <- paste0("https://fr.trustpilot.com", company_card %>%
  html_element("a") %>% html_attr("href"))

website <- rbind(website, data.frame(
  website_name = website_name,
  nb_avis = nb_avis,
  localisation = localisation,
  type = type,
  lien = lien
))
print(paste("page", j, "has been scraped"))

j<-j+1

}
return(website)

}

website<-trustpilot_website(url_start = url_start)

##### Création du fichier TP_pets_website.csv

```

```

website<-website%>%mutate(note=str_split_i(nb_avis,"\\|", 1)%>%
  str_remove_all(., "[A-z]")%>%
  str_replace(., ",", ".")%>%
  as.numeric(),
  nb_avis=str_split_i(nb_avis,"\\|",2)%>%
  str_remove_all(., "[A-z]")%>%
  str_remove_all(., "[:space:]")%>%
  as.numeric()%>%
  replace_na(.,0),
  url_start=url_start,
  nb_page=0,
  cat=str_split(type, ".")%>%
  unnest_wider(cat, names_sep = "_")

data_scrap<-website%>%filter(nb_avis>10)
write_csv(data_scrap, "TP_pets_website.csv")

### Fichier TP_pets_reviews.rds

##### Scraping reviews

trustpilot_reviews<-function(data){
  Sys.sleep(5)

  for (j in 1:nrow(data)) {
    i<-1
    b<-1

    while (b!="TRUE") {

      Sys.sleep(5)

      b<-http_error(paste0(data$lien[j], "?languages=all&page=", i))

      i<-i+1
      data$nb_page[j]<-i-2

    }
    print(paste0("nb_page of ", data$website_name[j], " has been fetched"))
  }
}

```



```

}

i<-1
reviews <- NULL
# cat("\014")
cat(paste0("The script will run on ", sum(data$nb_page), " pages!\n"))
Sys.sleep(5)

for (j in 1: nrow(data)){
  for (i in 1:data$nb_page[j]){
    url<-paste0(data$lien[j], "?languages=all&page=", i)
    Sys.sleep(5)
    session <- bow(url)
    session$user_agent<-"Googlebot"
    message("Scraping ", url)
    page<-nod(session, url) %>%
      scrape(verbose=TRUE)

    review_card <- page %>%
      html_elements("div.styles_reviewCardInner__EwDq2")

    name <- review_card %>%
      html_element("span.typography_heading-xxs__QKBS8.typography_appearance-default__AAY1")
      html_text()

    rating <- review_card %>%
      html_elements("div.star-rating_starRating__4rrcf.star-rating_medium__iN6Ty") %>%
      html_element("img") %>%
      html_attr("alt") %>%
      str_extract("[:digit:]")

    published <- review_card %>%
      html_elements(".styles_reviewContentwrapper__zH_9M") %>%
      html_element("p.typography_body-m__xgxZ_") %>%
      html_text() %>%
      str_remove("Date de l'expérience: ")

    verified <- review_card %>%
      html_element(".styles_detailsIcon__yqwWi") %>%
      html_text()
  }
}

```

```

title <- review_card %>%
  html_element("h2")%>%
  html_text()

content <- review_card%>%
  html_elements(".styles_reviewContentwrapper__zH_9M")%>%
  html_element("p.typography_body-1_KUYFJ") %>%
  html_text2()

reviews <- rbind(reviews, data.frame(
  website_name = data$website_name[j],
  name = name,
  rating = rating,
  published = published,
  verified = verified,
  title = title,
  content = content
))

i<-i+1
}
print(paste0(data$website_name[j], " has been scraped"))

j<-j+1
}

return(reviews)
}

hak<-trustpilot_reviews(data_scrap)

```

## References

- Balech, Sophie. 2022. “Une Application Du Modèle ELM ( Elaboration Likelihood Model ) Au Partage d’information Sur Twitter : Étude Du Rôle de La Forme Du Message Et Du Profil de l’émetteur.” *Innovations* n° 69 (3): 129–61. <https://doi.org/10.3917/inno.pr2.0135>.
- Balech, Sophie, and Christophe Benavent. 2022. “Le Rôle Des Dimensions de l’expérience Dans La Satisfaction Client : Une Application Au Cas de l’industrie Hôtelière En Polynésie Française.” In *38ème Congrès International de l’AFM*, 28–38. Tunis, Tunisie.
- Barron, Alexander T. J., Jenny Huang, Rebecca L. Spang, and Simon DeDeo. 2018. “Individuals, Institutions, and Innovation in the Debates of the French Revolution.” *Proceedings of the National Academy of Sciences* 115 (18): 4607–12. <https://doi.org/10.1073/pnas.1717729115>.
- Boegershausen, Johannes, Hannes Datta, Abhishek Borah, and Andrew T. Stephen. 2022. “Fields of Gold: Scraping Web Data for Marketing Insights.” *Journal of Marketing* 86 (5): 1–20. <https://doi.org/10.1177/00222429221100750>.
- Hartmann, Jochen, Mark Heitmann, Christina Schamp, and Oded Netzer. 2021. “The Power of Brand Selfies.” *Journal of Marketing Research* 58 (6): 1159–77. <https://doi.org/10.1177/00222437211037258>.
- Kozlowski, Austin C, Matt Taddy, and James A Evans. 2019. “The Geometry of Culture: Analyzing Meaning Through Word Embeddings.” *American Sociological Review* 89 (5): 769–981. <https://doi.org/10.1177/0003122419877135>.
- Kumar, Sunil, Arpan Kumar Kar, and P. Vigneswara Ilavarasan. 2021. “Applications of Text Mining in Services Management: A Systematic Literature Review.” *International Journal of Information Management Data Insights* 1 (1): 100008. <https://doi.org/10.1016/j.jjime.2021.100008>.
- Lefrançois, Alicia, Sophie Balech, and Sophie Changeur. 2022. “Transgression Et Consommation: Revue Intégrative Et Proposition d’un Agenda de Recherche.” In. Le Havre, France.
- Liu, Angela Xia, Yilin Li, and Sean Xin Xu. 2021. “Assessing the Unacquainted: Inferred Reviewer Personality and Review Helpfulness.” *MIS Quarterly* 45 (3): 1113–48. <https://doi.org/10.25300/MISQ/2021/14375>.
- Tirunillai, Seshadri, and Gerard J. Tellis. 2012. “Does Chatter Really Matter? Dynamics of User-Generated Content and Stock Performance.” *Marketing Science* 31 (2): 198–215. <https://doi.org/10.1287/mksc.1110.0682>.
- Vannoni, Matia. 2022. “A Political Economy Approach to the Grammar of Institutions: Theory and Methods.” *Policy Studies Journal* 50 (2): 453–71. <https://doi.org/10.1111/psj.12427>.